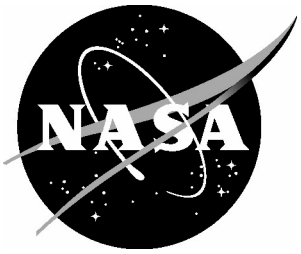
NASA/TM-2005-213540



ROBUS-2: A Fault-Tolerant Broadcast Communication System

Wilfredo Torres-Pomales, Mahyar R. Malekpour and Paul S. Miner
Langley Research Center, Hampton, Virginia

March 2005

The NASA STI Program Office . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the lead center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers, but having less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services that complement the STI Program Office's diverse offerings include creating custom thesauri, building customized databases, organizing and publishing research results ... even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at ***http://www.sti.nasa.gov***

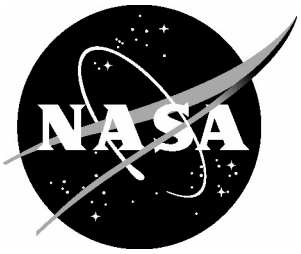
- E-mail your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA STI Help Desk at (301) 621-0134

- Phone the NASA STI Help Desk at (301) 621-0390

- Write to:
  NASA STI Help Desk
  NASA Center for AeroSpace Information
  7121 Standard Drive
  Hanover, MD 21076-1320

NASA/TM-2005-213540



ROBUS-2: A Fault-Tolerant Broadcast Communication System

Wilfredo Torres-Pomales, Mahyar R. Malekpour and Paul S. Miner
Langley Research Center, Hampton, Virginia

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-2199

March 2005

Acknowledgment

This work was supported, in part, by the FAA William J. Hughes Technical Center under interagency agreement DTFA03-96-X90001.

Available from:

NASA Center for AeroSpace Information (CASI)
7121 Standard Drive
Hanover, MD 21076-1320
(301) 621-0390

National Technical Information Service (NTIS)
5285 Port Royal Road
Springfield, VA 22161-2171
(703) 605-6000

Abstract

The Reliable Optical Bus (ROBUS) is the core communication system of the Scalable Processor-Independent Design for Enhanced Reliability (SPIDER), a general-purpose fault-tolerant integrated modular architecture currently under development at NASA Langley Research Center. The ROBUS is a time-division multiple access (TDMA) broadcast communication system with medium access control by means of time-indexed communication schedule. ROBUS-2 is a developmental version of the ROBUS providing guaranteed fault-tolerant services to the attached processing elements (PEs), in the presence of a bounded number of faults. These services include message broadcast (Byzantine Agreement), dynamic communication schedule update, clock synchronization, and distributed diagnosis (group membership). The ROBUS also features fault-tolerant startup and restart capabilities. ROBUS-2 is tolerant to internal as well as PE faults, and incorporates a dynamic self-reconfiguration capability driven by the internal diagnostic system. This version of the ROBUS is intended for laboratory experimentation and demonstrations of the capability to reintegrate failed nodes, dynamically update the communication schedule, and tolerate and recover from correlated transient faults.

Table of Contents

1. Introduction

The Scalable Processor-Independent Design for Enhanced Reliability (SPIDER) is a general-purpose distributed computer architecture currently under development at NASA Langley Research Center. The purpose of this effort is to design a flexible architecture that can be configured to satisfy a wide range of performance and reliability requirements, while preserving a consistent interface to application programs. One of the development goals is to develop the architecture such that it efficiently scales from a small configuration supporting a single aircraft function to a large distributed configuration performing multiple functions simultaneously. The architecture is expected to support functions of various criticality levels, including ultra-reliable and safety-critical aircraft functions with hard real-time deadlines.

SPIDER is designed as an integrated modular architecture (IMA) composed of a communication system and a set of processing elements (PEs). The Reliable Optical Bus (ROBUS) is a fault-tolerant, time-division multiple access (TDMA) broadcast communication system with medium access control by means of a time-indexed communication schedule. The ROBUS provides a set of basic communication services, and its essential goal is to ensure reliable communication between all pairs of fault-free PEs. The PEs perform two basic functions: execute the application software and run the distributed operating system (SPIDER-OS). The application-specific software executed by individual PEs may include processing of data, computing control functions, reading sensors, driving actuators, or providing a communication path to other networks (e.g., a gateway function). The SPIDER-OS handles the communication, process management, and redundancy management at the PE level. The SPIDER-OS consists of a commercial off-the-shelf (COTS) real-time operating system (RTOS) and a middleware layer located between the operating system and the application software. The SPIDER middleware provides an interface between applications running on the PEs and handles all the SPIDER-specific functions that are not a concern of application-specific software. The middleware enables the implementation of fault-tolerant strategies combining the PEs to provide fail-operational and fail stop capabilities in a way that is transparent to the application software. The redundancy management strategies at the PE level are flexible and can be adapted to support dissimilar processors.

The ROBUS is the central feature of SPIDER in the sense that it provides a set of basic services and guarantees upon which higher-level services are built. The approach selected for the development of SPIDER includes the design and implementation of concept demonstration versions of the ROBUS. Although it has fairly straightforward behavior at the external interfaces, internally the ROBUS is in fact a distributed system consisting of dedicated protocol processors that perform ROBUS-specific functions and are interconnected by a lower-level communication network. The developmental versions of the ROBUS will be leveraged in laboratory investigations to assess the effectiveness of the distributed protocols and the redundancy management strategies and to expose areas where further research and development is required. These demonstration versions of the ROBUS will also be used as test beds for the development of the SPIDER OS.

This document provides a description of ROBUS-2, an instance of the ROBUS designed to demonstrate the following bus capabilities: re-integration of repaired nodes, dynamic update of the communication schedule, and fault-tolerance and recovery from correlated transient faults. This instance of the ROBUS also serves as a design case for the study of robustness and efficiency in implementations of the error detection, diagnosis, and reconfiguration strategies developed up to this point. In addition, ROBUS-2 is intended to demonstrate that the bus can achieve a PE-message throughput that approaches the available bandwidth at the physical communication layer, while preserving the fault-tolerance guarantees.

The first version of the bus, ROBUS-1, is described in [Miner 02]. The design of ROBUS-2 is based on the unified fault-tolerance protocol discovered by Miner, et al [Miner 04]. That protocol is a generalization and extension of the Byzantine fault-tolerance protocol introduced by Davies and Wakerly [Davies 78].

[Rushby 03] presents a comparison of bus architectures for safety-critical applications, including SAFEbus, TTA, FlexRay, and SPIDER.

1.1. Basic services

ROBUS-2 provides four basic fault-tolerant services.

- **Message broadcast**: Every scheduled message sent by a PE is delivered to all of the properly working PEs. Irrespective of the status of the source PE, all of the properly working PEs will agree on the content of each message. If the source PE is working properly, all of its messages will be received exactly as they are sent.

- **Communication schedule update**: The PEs can dynamically modify the bus access pattern by downloading a new communication schedule to the ROBUS.

- **Time reference**: ROBUS-2 provides an accurate and precise time reference to the PEs, which they can use to coordinate their actions.

- **Self-diagnosis**: ROBUS-2 can detect and diagnose internal failures with a high degree of coverage. Diagnosed component failures are periodically reported to the PEs so they can react appropriately according to their application.

1.2. Additional features

Other features of ROBUS-2 include the following.

- **Time-triggered operation**: Normal activity on the bus is controlled by time-indexed internal operation schedules that specify exactly when to begin the processing for each service and, for most protocols, exactly when to start all the transmissions. In addition, a highly effective fault-tolerant time synchronization protocol enables the bus to measure time with fine resolution. These are critical elements that give the bus the ability to deliver services with predictable timing, even in the presence of faults.

- **Communication schedule enforcement**: ROBUS-2 grants access to the bus only as indicated by the communication schedule. The enforcement mechanism ensures that faulty PEs do not interfere with other PEs accessing the bus.

- **Self-reconfiguration**: Internal error detection and diagnosis allows ROBUS-2 to quickly identify and neutralize failed internal components. These mechanisms also allow the bus to re-integrate repaired components.

- **Internal-fault masking**: ROBUS-2 incorporates a fault-masking capability that allows it to tolerate a

bounded number of undiagnosed internal component failures.

- **Fault-tolerant startup and restart**: The error handling mechanisms are active during initialization. This enables the bus to start up with variable initial configurations and in the presence of component failures. In addition, the error handling mechanisms enable ROBUS-2 to detect many transient errors and take appropriate actions to clear and re-integrate the affected components. These mechanisms coupled with the startup capability give ROBUS-2 the means to recover from some scenarios of massive transient faults affecting the system.

- **PE-fault tolerance**: ROBUS-2 design allows it to maintain internal coordination and continue service delivery independently of the number of failed PEs. Error detection applied to the communication schedule updates enables the detection of invalid schedules, in which case ROBUS-2 activates a default schedule to ensure that the PEs can continue to communicate.

This version of the ROBUS is intended for implementations with a relatively small number of PEs, say fewer than seven. Future versions will include various design optimizations to enable efficient implementations with a much larger number of PEs.

1.3. Document organization

This document is intended to be a comprehensive and self-contained design reference including description and analysis. The following sections describe the design of ROBUS-2 in detail. The presentation begins with an overview of the behavior and structure of the bus. This is followed by a description of the message format and the distributed coordination strategy for the implementation of the ROBUS-2 protocols. The diagnostic system, including the diagnostic policy, is described. Then, the modes of operation of the bus are presented, including descriptions for each of the protocols. The appendices present relevant background concepts and the basic theory of fault tolerance and communication, as well as analysis for the ROBUS-2 protocols and the startup and restart capability. Throughout, the document provides insight into the operation of the design, including how to set up critical aspects of the system for an actual physical implementation.

From this point on, we refer to the bus described here simply as “ROBUS”. It should be understood everywhere, unless explicitly stated otherwise, that we are referring to the ROBUS-2 version of the bus, and not about ROBUS in general.

2. System overview

The following introduces the design of the ROBUS and serves as an overall reference for later sections, which cover particular design elements in detail.

2.1. System behavior

This section presents a brief overview of the behavior of the ROBUS.

2.1.1. Basic states

Figure 2.1 shows a simplified view of the high-level state transitions. The bus is deactivated by cutting off power to the system. When enabled, it executes an initialization routine and then proceeds to begin service delivery. The bus will remain engaged until it is deactivated or a failure condition is detected. If a failure occurs, the bus will try to re-establish service delivery as soon as possible. For ROBUS-2, all bus failures are presumed to be transient. Thus, the bus is designed to never give up trying to return to normal operation.

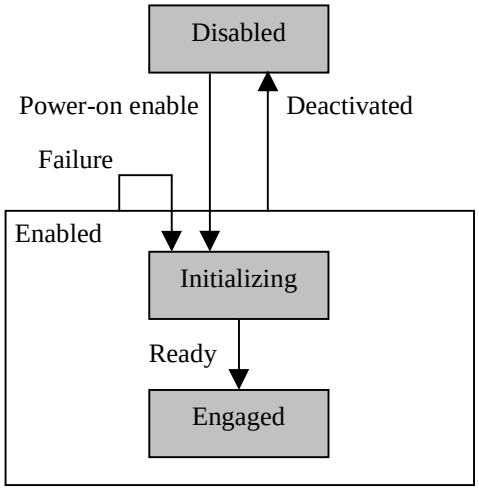


Figure 2.1: Simplified high-level state-transition graph for ROBUS

2.1.2. Steady-state operation

The steady-state behavior of the ROBUS consists of a simple cyclic operation. As illustrated in Figure 2.2, in each cycle the bus goes through a predetermined sequence of protocols to deliver the expected services: time reference, self-diagnosis, communication schedule update, and PE message broadcast. Note that Figure 2.2 is not drawn to scale. Most of the time in a cycle (say, over 90%) is available for the broadcast service.

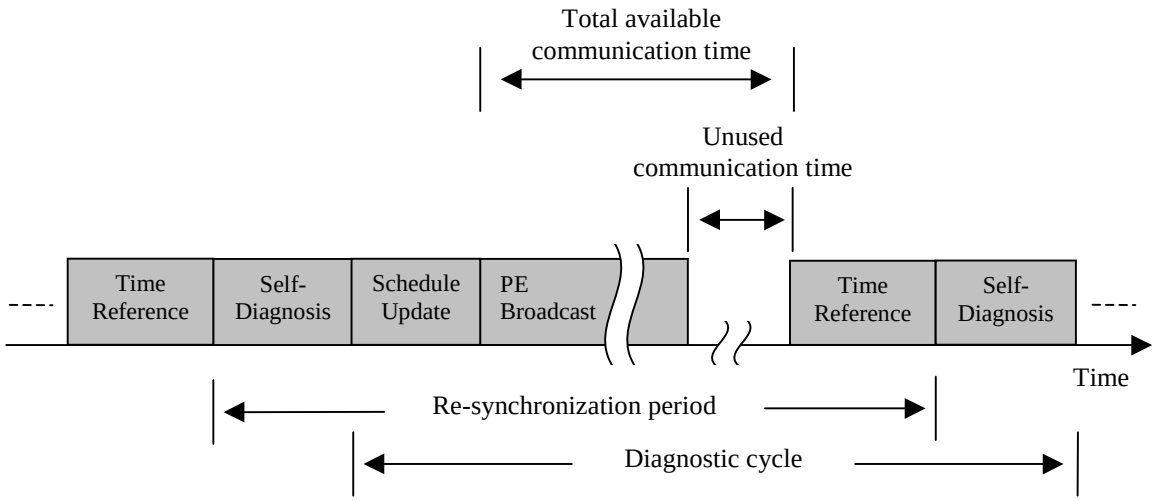


Figure 2.2: Service delivery sequence

The Time Reference service provides a periodic time update in the form of a dedicated message simultaneously broadcast from the bus to the PEs. The period between updates, called the **re-synchronization period**, is nominally specified before run time. The time reference indicates the time kept by the bus, which is not synchronized to an external time source. (The PEs can maintain dedicated time clocks synchronized to an external time reference independently from the ROBUS time service. Those clocks would be updated periodically with adjustments agreed to by the PEs using an agreement protocol and communication via ROBUS.)

During Self-Diagnosis, the bus sends out to the PEs the latest available results of internal diagnosis. The interval from the end of one self-diagnosis to the end of the next is called a **diagnostic cycle**. The protocol used for this service ensures that the PEs receive consistent diagnostic information. This information can be used by the PEs for process and redundancy management decisions at the SPIDER level.

During Schedule Update, all the PEs simultaneously send their desired schedule to the bus. The schedule specifies the number of messages that will be transmitted by each PE during the next broadcast service. Ideally, all the PEs agree on the communication schedule before it is sent to the ROBUS. However, the ROBUS is designed to tolerate a condition in which there is no agreement among the PEs. This is accomplished by using error detection and an agreement generation protocol. If the ROBUS detects that the received schedule is invalid, it will reject it and a default schedule will be used. The final decision on the schedule to be used is forwarded back to the PEs.

In PE Broadcast, the ROBUS grants bus access to individual PEs according to the communication schedule. An interactive consistency protocol is used for each scheduled message to ensure that the PEs receive consistent messages. The bus access pattern is a time-indexed, as-soon-as-possible (ASAP) round-robin sequence. Figure 2.3 provides an example of the access pattern. The PEs are identified according to the statically assigned identification numbers which uniquely identify each ROBUS port. The PEs access the bus in ascending order according to the port identification numbers. The first scheduled message is sent at some predetermined time. The interval between the send time of one message and the send time of the next (known as the **data introduction interval** or DII) [De Micheli 94] is constant. After all the scheduled messages for one PE have been sent, the messages for the next PE are

broadcast maintaining the DII between messages. If one PE is not scheduled to send messages, then the messages for the next scheduled PE are sent. After all of the scheduled messages are processed, the bus remains idle until the time to restart the Time Reference service.

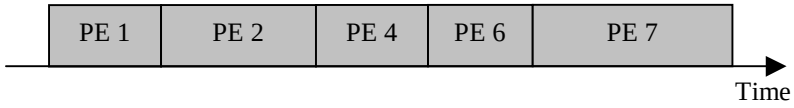


Figure 2.3: Example of an access pattern during the PE Broadcast service

2.2. System structure

Figure 2.4 shows the ROBUS topology. The bus has an active star architecture with the **Bus Interface Units** (BIUs) serving as the bus access ports and the **Redundancy Management Units** (RMUs) providing connectivity as network hubs. The network between BIUs and RMUs forms a complete bipartite graph in which each node is directly connected to every node of the opposite kind. Only the links shown are available for communication. There are no functional links between BIUs or between RMUs, and the RMUs have no direct links to the external world. All of the communication links are bidirectional. The design of the ROBUS is independent of the physical point-to-point communication technology and is suitable for use with point-to-point optical data links.

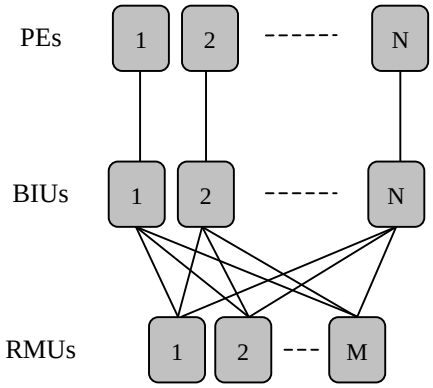


Figure 2.4: ROBUS topology

The number of BIUs, denoted by N, is fixed. The number of RMUs is denoted by M and is also fixed. Every BIU is assigned a unique node identification number from 1 to N. Likewise, the RMUs are assigned numbers from 1 and M. Each PE is uniquely identified by the number of its corresponding BIU.

Using Figure 2.4 it is easy to see how the communication schedule can be enforced. Since the PEs are connected to the bus via the BIUs, it is the responsibility of each BIU to ensure that the messages from its attached PE are forwarded to the RMUs only at allowed times. Similarly, since the BIUs are attached directly to the RMUs, the RMUs are responsible for ensuring that only the messages from the scheduled BIU (and its corresponding PE) are relayed back to the BIUs. The most important aspect of the bus-access enforcement mechanism is to control access the RMU-to-BIU links.

2.3. Node behavior

Figure 2.5 presents a simplified view of the high-level state-transition graph for the ROBUS nodes. BIUs and RMUs. Both BIUs and RMUs follow this same pattern of behavior. This graph is essentially the same as the one for the ROBUS shown in Figure 2.1. In the Disabled state, a node is powered off or otherwise removed from active bus participation. Once enabled, a node enters the Initializing state where it tries to find other nodes suitable for providing communication services to the PEs. Once a node has confirmed that it is operating in a proper configuration with other nodes, it enters the Engaged state. To deliver services to the PEs, it is necessary for a group of BIUs and RMUs to work together in a coordinated way. We refer to a group of BIUs and RMUs that can be relied upon to deliver proper services to the PEs as a **clique**. An initializing node becomes engaged after it identifies a clique and becomes part of it. If a node determines that a significant failure condition is present while being part of clique, the node transitions back to the Initializing state to reset its state and attempt to re-engage. A ROBUS node can be designed with the capability to transition to the Disabled state when it determines that it cannot form or join a clique due to local permanent faults or some condition that is outside the recovery capabilities and is interpreted as a permanent failure. That feature, illustrated by the dashed arrow, is not included in ROBUS-2.

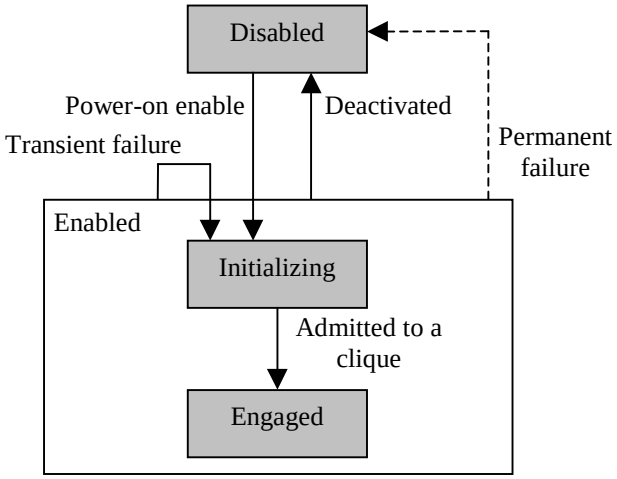


Figure 2.5: Simplified high-level state-transition graph for BIUs and RMUs

2.4. Node structure

Figure 2.6 depicts the basic structural components of a ROBUS node. This decomposition applies to BIUs and RMUs. The Communication Module handles all the point-to-point communication and uses mostly commercial off-the-shelf (COTS) components. The links between BIUs and RMUs implement broadcast communication using either one-to-one or one-to-many links. If the BIUs and the PEs are physically separate (see the topic **Fault containment** in a later section), the interconnection between them must use one-to-one links. If they are not separate, then some other means for local data transfer can be used.

The Computation Module, also known as a ROBUS Protocol Processor (RPP), handles all the ROBUS-specific functions including mode transition logic, low-level protocols, error detection, diagnosis, reconfiguration, and distributed coordination.

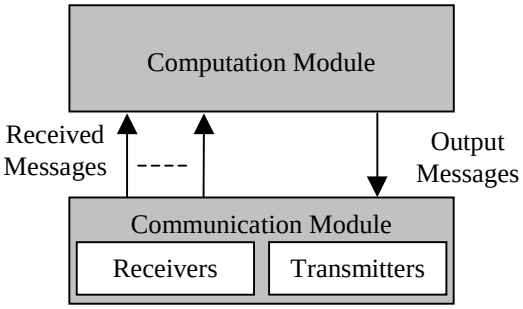


Figure 2.6: Generic node structure for BIUs and RMUs

2.5. Distributed coordination

Each ROBUS node is driven by an independent, free-running physical oscillator. These oscillators are characterized by a known bound on their drift rate with respect to real time. Each node also has a logical-time clock, referred to as the **local-time clock**, which keeps track of the passage of time as indicated by the physical oscillator. Given an initial precision of synchronization for the local times at any two nodes, the precision can worsen over time at a rate determined by the drift rate of the physical oscillators.

The ROBUS protocols are divided into two categories: synchronization protocols and synchronous protocols. The **synchronization protocols** use event-triggered communication and event-processing operations to generate high-precision distributed events that are used to synchronize the local-time clocks. The **synchronous protocols** use time-triggered communication and operations in order to process information. To achieve proper coordinated action in the execution of the synchronous protocols, the local-time clocks of the participating nodes must be synchronized within some known bounded precision.

The ROBUS has two synchronization states: synchronized and unsynchronized. In the **synchronized** state, the precision of synchronization is determined by an internal distributed reference event generated by a clock synchronization protocol. The precision of this event allows the nodes to achieve very tight local-time synchronization. The bus is in the **unsynchronized** state when it transitions to the startup and restart processes. The precision of synchronization in this state is mainly determined by events not directly controlled by the bus. It is assumed that the synchronization precision in this mode has a known bound that can be large relative to the precision in the synchronized state. The bus transitions from the unsynchronized state to the synchronized state after the execution of a synchronization initialization protocol. Because the local times can drift apart, the synchronization protocol must be re-executed at regular intervals to ensure that the local times are kept synchronized. The rate of re-synchronization is constrained by physical parameters of the design (e.g., oscillator drift rates) as well as precision and accuracy goals. The fault-tolerance attribute of the synchronization protocol enables the bus to maintain synchronization even in the presence of failed nodes.

Startup and restart of the bus are particularly difficult scenarios to handle properly, especially in the presence of arbitrary faults. The ROBUS achieves synchronization during startup and restart by exploiting the properties of the initial synchronization protocol. With this protocol, the ROBUS can synchronize if the nodes start within a known bound of the relative local-time skew. The critical property concerning this capability of the synchronization protocol is that, although the initial relative skews must

have a known bound, this bound can be arbitrarily large. This feature enables the use of physical events beyond the sphere of control of the nodes as distributed reference events to coordinate the startup and restart processes. The local power-on enable, which is externally controlled by the system user, is used by the bus as a reference event for startup. The detection of a bus failure, which is triggered by some fault-causing phenomenon, is used as a reference event for restart. The worst-case precision of these events determines the bound on the initial relative local-time skew in the unsynchronized state.

The execution of synchronous protocols is driven by the local time and a time-indexed operation schedule. The low-level distributed protocols specify the node activities by defining the operations, the operation sequencing, the message flow patterns, and the executing nodes for each operation. The timing of the operations is determined using a model of distributed synchronous composition. This execution scheme and the high synchronization precision in the synchronous state make the steady-state behavior of the ROBUS highly deterministic as it precisely specifies the timing of all the internal communication between BIUs and RMUs, as well as the communication with the PEs. The concept of distributed synchronous composition is explained in detail in Section 3.

2.6. Redundancy management

The purpose of redundancy management is to increase the probability of continued service delivery through effective utilization of available resources. The ROBUS is designed to manage its redundant BIU and RMU components independent of the PEs.

2.6.1. Fault containment

Fault containment refers to the confinement of physical faults to a limited locality. This is achieved by establishing containment boundaries defined by fault propagation barriers that prevent faults from spreading indiscriminately throughout the system. Each area enclosed by containment boundaries is known as a **fault containment region** (FCR) (see [Lala 91]). Ideally, the FCRs are independent from each other in the sense that physical faults in one FCR will not cause faults in others. Communication between FCRs is through carefully specified interfaces that ensure a sufficiently high degree of fault containment. Fault containment is a fundamental requirement of most fault-tolerant systems. In [Driscoll 03], Driscoll, et al present a particularly devious fault propagation mechanism that can wreak havoc in a system if not properly addressed in the design of the FCRs. For ROBUS, every BIU and RMU node is in a separate FCR. Each BIU can be by itself in a FCR, or it can share an FCR with its attached PE.

Although FCRs can prevent the propagation of faults, they do not preclude the simultaneous presence of physical faults in separate FCRs caused by independent phenomena internal to the system. In addition, external threats like lightning and high-intensity radiated fields (HIRF) have the potential to disturb multiple FCRs. It is presumed that the fault-containment solution does not prevent the propagation of environment-induced faults within a FCR. Therefore, when a fault is detected in a FCR, all the components within the FCR are presumed to be affected and no specific assumptions are made about the behavior of the corresponding ROBUS node. The ROBUS is designed with mechanisms that can handle a large number of coincident faults and arbitrary fault manifestations.

2.6.2. Error detection

Error detection is based on the comparison of actual attributes of observed data against expected

attributes. The ROBUS nodes use six categories of error detectors. These checks generate syndromes that are used to diagnose the system.

- **Communication checks:** Each communication link should have a high-coverage error-detection capability for errors occurring anywhere from the transmitter to the receiver.

- **In-line checks:** These checks individually compare received messages against expected characteristics of timing and content.

- **Cross-lane checks:** These checks compare received messages against the result of a vote. The checks are performed on timing and content characteristics.

- **Protocol checks:** These checks are essentially sanity checks on intermediate and/or final protocol results based on expected behavioral characteristics of the ROBUS.

- **Self-checks:** These checks are performed by a node to monitor its own operation. The self-checks described in this document are based on properties of the ROBUS protocols. Other protocol-independent or application-specific checks can be defined to increase the error coverage.

- **PE-error checks:** These checks are not specified in this document. However, the system is designed to accept and process error syndromes about expected PE messages at the BIUs.

2.6.3. Diagnosis

Each BIU and RMU node is an **observer** of every node. An observed node is known as a **defendant**. A **direct observer** receives information from the defendant by way of a direct data communication link. An **indirect observer** receives information from the defendant by way of direct observers. Due to the ROBUS topology, a node is a direct observer of nodes of the opposite kind and an indirect observer of nodes of the same kind, including itself. Every ROBUS node is a defendant and an observer. The purpose of diagnosis is to assess the status of each node and the bus as a whole. The diagnostic system of the ROBUS is a distributed system divided into two layers. In the local layer, the nodes monitor the communication and independently diagnose each node and the bus. In the collective layer, the nodes exchange diagnostic information to augment their local diagnoses.

The diagnostic system assesses each node to determine its suitability to participate in the delivery of services to the PEs. A **trustworthy** node can be relied upon to deliver the expected services. **Untrustworthy** nodes do not behave as expected and, thus, are sources of errors. The causes of errors by a node can be physical defects or disturbances, or incorrect values held in the state variables.

There are three steps to diagnose a node: error detection, culprit identification, and assessment. The error checks of the types described in the previous section are used to generate error syndromes. Error sources are identified using the error syndromes and knowledge of the protocols and the topology. Some error syndromes unequivocally point to a single error source, while others are ambiguous and require the combination of multiple syndromes in order to locate the error source. The diagnostic system uses a local hierarchical classification scheme and policy-based rules to assess the status of each node. Each step in the hierarchy corresponds to an increase in the severity of the assessment. A node is **suspected** by an observer when it determines that the defendant is one of several possible culprits for a detected error. A node is **blamed** when an observer determines that the defendant is a source of detected errors. A node is

accused by an observer when it determines that the defendant is untrustworthy, but is uncertain whether other observers have reached the same conclusion. A node is **convicted** when the observers agree that a sufficient number of them consider the defendant untrustworthy. For this version of the ROBUS, each node uses Boolean variables for all the diagnostic information.

The BIU and RMU members of a clique work together in a coordinated way to deliver services to the PEs. A clique is considered trustworthy if it is suitable to deliver services according to the specification. The diagnosis of the bus consists of determining if a trustworthy clique is in operation. For this version of the ROBUS, it is assumed that at any time there is at most one stable trustworthy clique on the bus. The diagnostic system uses error syndromes, knowledge of the protocols, the results of node diagnostics, and policy-based rules to assess the status of the bus.

2.6.4. Reconfiguration

The purpose of reconfiguration is to enhance the ability of a clique to establish and preserve proper service delivery in the presence of untrustworthy nodes. The membership of a clique is determined using the results of diagnosis. A clique is reconfigured by adding or removing nodes from its membership. A member of a clique is allowed to participate in the delivery of services to the PEs and is referred to as a **trusted** node. We refer to a node searching for or trying to become part of a clique as a **recovering node**.

The reconfiguration strategy of the ROBUS is driven by the need to handle scenarios with a large number of simultaneous or nearly simultaneous node failures caused by harsh environmental phenomena. Although the ROBUS has the capability to re-initialize a failed clique, the preferred way to handle a fast increase in the number of untrustworthy nodes is to preserve the delivery of services by quickly removing as many untrustworthy nodes as possible. The presence of a surviving clique forces recovering nodes to execute a re-integration procedure to rejoin the clique. The re-integration procedure of the ROBUS is considered more robust than the re-initialization procedure, which has strict assumptions about the duration of the fault-causing phenomenon and the failure detection delay. In addition, the coordinated and highly deterministic activity of a clique engaged in service delivery to the PEs enables the application of detailed error detection and diagnosis by the recovering nodes and the clique. This allows the expansion of the clique to proceed with a high level of protection against untrustworthy nodes. Another advantage of preserving a degraded clique is that it increases the likelihood that at least some PEs can continue to do useful work.

2.6.5. Error containment

Error containment refers to the establishment of barriers to prevent incorrect information from propagating throughout the system. The error propagation barriers define partitions called **error containment regions** (ECRs). Similarly to the FCRs, every BIU and RMU is in a separate ECR. Also, each BIU can be by itself in an ECR, or it can share an ECR with its attached PE.

The only error propagation path between ECRs is through their interfaces. Thus, error containment can be achieved by placing barriers at one or both ends of each interface. The effectiveness of an error propagation barrier, referred to as the **error-containment coverage**, is measured by the probability that errors will not propagate across the barrier. For the interfaces between BIUs and RMUs, error containment is realized by a fail-stop mechanism to block errors at the source end of an interface, and input error detection and voting to block errors at the receive end. The use of error propagation barriers between BIUs and PEs is optional and their definition is not part of this document.

2.6.5.1. Fail-stop nodes

The goal of fail-stop behavior is to increase the error-containment coverage of an interface. Errors at a source node can affect output transmissions in unknown ways. Fail-stop behavior prevents the indiscriminate propagation of errors out from an ECR by mapping detect failures to a condition of no output activity, which can be consistently identified by the nodes at the receiving end as an indication of an untrustworthy source.

The ROBUS nodes disable their output ports as soon as a local failure or a bus failure is detected. These conditions are indications that a node should not continue with normal activity because its transmissions are likely to be erroneous or the receiving nodes are not operating properly.

The fail-stop reaction of the ROBUS nodes is not permanent. As mentioned in a previous section, the nodes in this version of the ROBUS do not implement a transition to a disabled state. Instead, following a failure, the nodes always try to recover and re-enable their outputs as required by the recovery procedures.

2.6.5.2. Input error detection

Input error detection prevents errors from entering an ECR. The location of detectors at the receiving end of an interface allows them to provide coverage for errors originating at the transmission source or somewhere in the communication path from the source to the receiver. In the ROBUS, input error detection is realized by the communication and in-line checks.

2.6.5.3. Dynamic voting

Most ROBUS operations involve redundant sources and voting performed at the receivers to reduce the information to a single result. As for the case of input error detection, voting at the receiving end of an interface provides protection against errors originating at a transmission source or in a communication path. The voting operations used by the ROBUS fall under the general category of dynamic voting, in which only a selected group of inputs is considered in the voting operation. The sources whose inputs are allowed to participate are called the **eligible voters**. The selection of eligible voters is based on the available results of node diagnosis and error detection performed on the inputs. Dynamic voting enables the ROBUS to quickly apply diagnostic results in order to enhance error containment and is the foundation of the internal-fault-masking feature of the bus. Three types of voting operations are defined for this version of the ROBUS: middle-value-select event voting, exact-match majority word voting, and exact-match majority bit voting.

Middle-value-select event voting is the basic operation used by the synchronization protocols to process timing events. In these protocols, the voting function, referred to as the **Accept** function, produces an output a fixed delay after it receives the middle event from the eligible voters. Let E denote the number of eligible voters. The middle event is defined as event number $\lceil (E + 1)/2 \rceil$. Equivalently, the middle event is the first event after $\lfloor E/2 \rfloor$ events have been received.

The unit of data for exact-match majority word voting is the multi-bit word. For this operation, referred to as a **word vote,** there is an exact-match majority among the input eligible voters if at least $\lceil (E + 1)/2 \rceil$ of the input words are exactly equal. Two eligible inputs are equal if they are an exact match in a bit-by-bit comparison. If there is a majority, the result of the vote is equal to the majority word. Otherwise, the result is undetermined and a no-majority condition is asserted.

The unit of data for exact-match majority bit voting is the bit. This operation, called a **bit vote**, is used for processing Boolean diagnostic variables like suspicions, accusations, and convictions. For this function there is an exact-match majority if at least $\lceil (E + 1)/2 \rceil$ of the eligible input bits are equal. If a majority of the eligible inputs are FALSE, the result is FALSE. Otherwise, the result is TRUE. This function definition is biased against the defendant. (This bias is justified by the analysis in Appendix F.)

2.7. Major operational modes

Figure 2.7 presents the mode transition graph for the ROBUS nodes. This graph applies to BIUs and RMUs. After a power-on enable, a node goes to the **Self-Test** major mode to perform a local initialization and test its circuitry. The node will remain in this mode indefinitely unless it successfully passes the test. After exiting this mode, the node proceeds to determine the status of the bus.

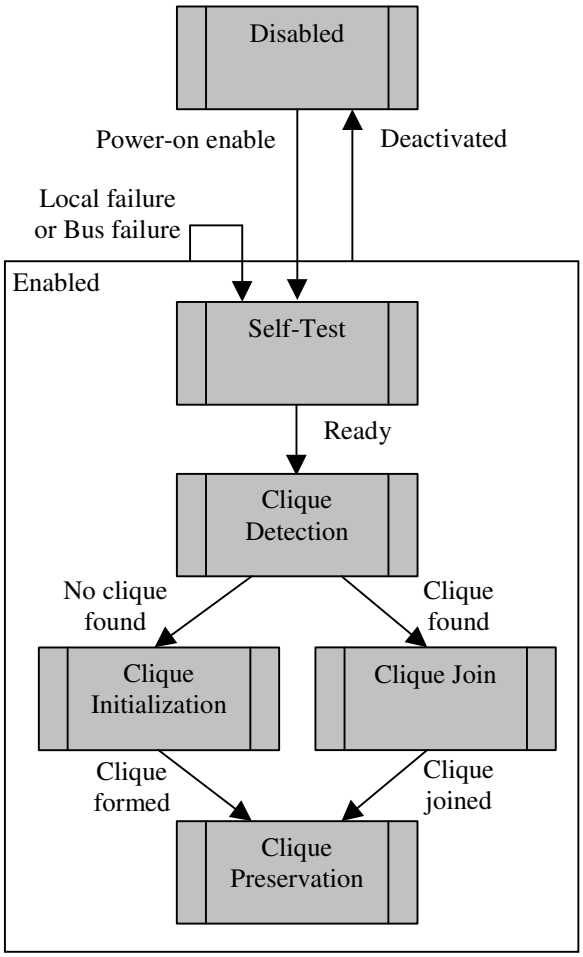


Figure 2.7: Major operational mode transitions for ROBUS nodes

The **Clique Detection** major mode consists of three minor modes. In **Local Diagnosis Acquisition,** a node uses unsynchronized local observations to make a first assessment of the likely members of a clique. In **Synchronization Acquisition**, the node attempts to synchronize to the clique. In **Collective Diagnosis**

Acquisition, the node captures the health assessment for each node as determined by the clique during the execution of the distributed diagnosis protocol. If at any time during the Clique Detection mode the node determines that no clique is present, it will exit this mode and attempt to form a new clique. Otherwise, it will assume that a clique exists and will try to join it.

A node transitions to the **Clique Initialization** major mode to form a new clique. The first minor mode is **Initial Diagnosis**, in which a node identifies other nodes also attempting to form a new clique. This is followed by the **Initial Synchronization** and **Collective Diagnosis** minor modes, where the nodes are synchronized and a consistent clique membership is established.

When a node enters the **Clique Join** mode, its state is in agreement with the state of the clique. In this mode, the node runs for two diagnostic cycles, essentially trying to demonstrate that it can be trusted. The existing members of the clique will integrate the node as soon as they confirm that the admission rules have been satisfied.

In the **Clique Preservation** major mode, a clique delivers services to the PEs according to the operation schedule. In the **Schedule Update** minor mode, a schedule-download protocol is executed to allow the PEs to reprogram the bus according to their communication needs. During **PE Communication**, first the PE messages are broadcast according to the communication schedule, and then the BIUs and RMUs exchange accumulated accusations against nodes of the opposite kind, which serves to enhance the diagnosis and reconfiguration capabilities of the bus. This is followed by a re-synchronization of the local time in the **Synchronization Preservation** mode and then a reassessment of the clique membership in the **Collective Diagnosis** mode.

2.8. Startup and restart

The ROBUS has a flexible capability to set up a clique and change its membership using the reconfiguration mechanisms. These mechanisms do not have restrictions on the number of nodes that can be simultaneously removed or admitted to a clique. As long as the clique membership is not overrun by untrustworthy nodes, the trustworthy nodes will be able to continue service delivery.

To start up a disabled bus, a group of BIUs and RMUs must be enabled within a known bounded time interval. Since there is no clique present, the nodes will reach the Clique Initialization mode and then transition to the Clique Preservation mode. The size of this initial clique can range anywhere from one BIU and one RMU to all BIUs and RMUs. Subsequently enabled nodes, if there are any, will detect the existing clique and follow the Clique Join path to be integrated into the clique.

A node determines that a local failure has occurred when its self-check detectors are triggered or when it is removed from the membership of a clique. In this case, the node transitions to the Self-Test mode, and then it attempts to re-integrate into the clique.

The nodes detect a clique failure when not enough BIUs and RMUs are trusted, and when the results of collective operations do not satisfy expected characteristics. It is possible for a clique in steady-state operation to recover from massive coincident transient faults that overwhelm its degradation and fault-tolerance capabilities. The re-initialization scheme assumes that the worst-case duration of a transient fault-causing phenomenon and the delay to detect the bus failure can be bounded. This is used to determine a bound on the initial relative local-time skew when entering the Clique Initialization mode.

Although highly unlikely, it is theoretically possible for coincident transient faults to corrupt the system in such a way that the nodes are divided into multiple mutually exclusive cliques simultaneously operating on the bus. In general, the ROBUS does not have the capability to recover from such conditions.

3. Communication and distributed coordination

This section describes the mechanism for communication between BIUs and RMUs, and the approach used to coordinate their activities. The communication between PEs and BIUs is also described, including the general data transfer model used at the BIU interface.

3.1. ROBUS Messages

The unit of data transfer in the ROBUS is the **ROBUS Message** (RM). As shown in Figure 3.1, a ROBUS message is composed of a one-bit **Tag** field followed by a fixed-size **Payload** field. This basic format is used for all the protocols. The Tag field has one of two values: SPECIAL or DATA. The relation between the Tag field value and the corresponding bit value on the message is implementation-dependent. The format and content of the Payload field depends on the value of the Tag field and the context in which the message is used.

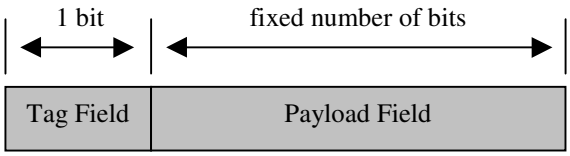


Figure 3.1: ROBUS message format

If the Tag field is SPECIAL, then the Payload field carries a bit pattern corresponding to one of the following labels.

- SELF_TEST
- CLIQUE_DETECTION
- CLIQUE_INITIALIZATION
- CLIQUE_JOIN
- CLIQUE_PRESERVATION
- VALID_SCHEDULE
- ZERO_SCHEDULE
- INVALID_SCHEDULE
- INIT
- ECHO
- PE_ERROR
- SOURCE_ERROR
- NO_MAJORITY

S_{RM} denotes the number of SPECIAL ROBUS messages. The assignment of bit patterns to the Payload labels is an implementation decision. The listed labels are a collection of all the SPECIAL messages defined for this version of the ROBUS. The interpretation of each label is dependent on the context in which the message is used.

If the Tag field is DATA, then the Payload field carries data with a format and content specific to the context in which the message is used. Three minor modes use DATA messages: Collective Diagnosis, Schedule Update, and PE Communication.

For Collective Diagnosis, the Payload field of each ROBUS message carries diagnostic data in the form of a Boolean vector. Figure 3.2 illustrates the format of diagnostic messages for the case of D defendants. Element b_i denotes a Boolean variable corresponding to an accusation or conviction against defendant i, which can be a BIU or an RMU. If the diagnosed defendants are BIUs, then D equals the number of BIUs, which is denoted by N. Otherwise, D is equal to the number of RMUs, denoted by M. The assignment of value to any unused bits is implementation-dependent.

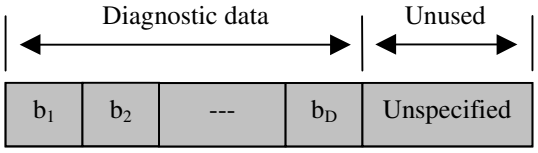


Figure 3.2: Payload format for diagnostic ROBUS messages

For Schedule Update, the DATA messages carry the number of messages scheduled for a particular PE. For this version of the ROBUS, it is valid to schedule a single PE to source the maximum number of messages that the bus can send during PE Communication, which is denoted by $K_{PE}|_{max}$. Therefore, the Payload field for Schedule Update DATA messages corresponds to an integer in the range 0 to $K_{PE}|_{max}$.

For the PE Broadcast protocol in the PE Communication mode, the DATA messages carry information from the PEs. The format of these messages is application-dependent. L_{PE} denotes the minimum Payload width requirement for PE messages. The exchange of accusations after the completion of the scheduled broadcasts uses the payload format for diagnostic ROBUS messages.

In addition to the protocols mentioned above, each BIU uses a DATA message to send its identification number to its attached PE. The Payload field for these messages corresponds to an integer number in the range of 1 to N.

The width of the Payload field, denoted by L_{PF}, must satisfy the following constraint.

$$L_{PF} \geq \max(\lceil \log_2(S_{RM}) \rceil, N, M, \lceil \log_2(K_{PE}|_{max} + 1) \rceil, L_{PE}) \quad (3.1)$$

3.2. Node process model

Figure 3.3 illustrates the process decomposition for the Computation Module of the ROBUS nodes. The Mode, Local Time, Diagnostics, and Schedule Processes hold the state information of the node. The Receive, Computation, and Send Processes perform protocol-specific operations. The Computation Process handles all the computation required by the protocols. The Send and Receive Processes interface with the local Communication Module and handle the ROBUS-specific communication functions. For the BIUs, the PE Interface handles the communication with the PEs. Error checks are located throughout the processes as appropriate. The timing patterns of the processes vary depending on the protocol being executed.

3.3. Communication between BIUs and RMUs

The ROBUS requires bidirectional communication between each BIU and RMU pair. This is realized using independent communication links in each direction. The communication links must provide adequate protection against the propagation of physical faults between interconnected nodes.

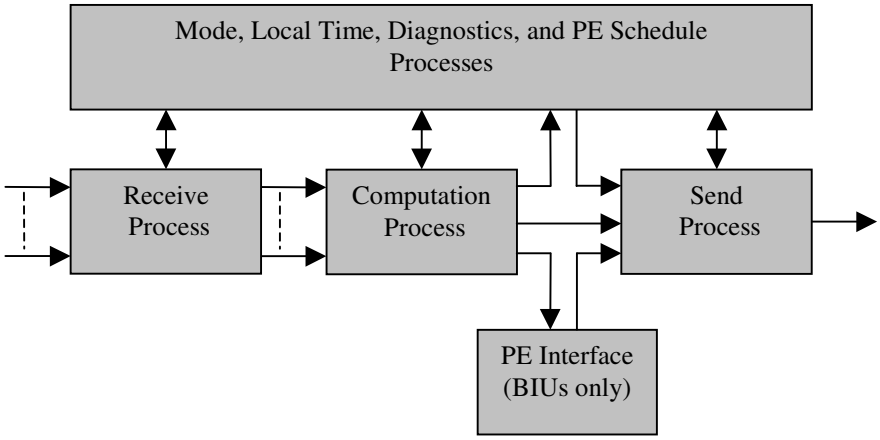


Figure 3.3: Main processes for ROBUS nodes

The behavioral design of the ROBUS requires that every node is able to broadcast ROBUS messages to the nodes of the opposite kind, simultaneously transmit and receive messages, and independently receive messages from every node of the opposite kind. The broadcast transmission function can be implemented using one-to-one or one-to-many transmitters. The reception requirements for BIUs and RMUs are satisfied by having a separate and independent receiver for each node of the opposite kind. In order to limit the cost and complexity of the system, the communication resources use mostly COTS components and every node uses the same communication links for synchronization and synchronous protocols. Contention in accessing the communication resources is prevented by the proper scheduling of the protocols and their operations.

The operation of the links is characterized by the transmission delay and the throughput. The **delivery delay** for a point-to-point link is the real time elapsed from the instant a ROBUS message is input to the transmitter until it is output at the receiver. The **delivery precision** is the range of variation of the delivery delay for a broadcast transmission. The throughput of a link is measured in terms of the minimum **data introduction interval** (or DII) [De Micheli 94], which is the minimum time required by the link between the instants at which consecutive messages are input to the transmitter. In general, smaller values of the delivery delay, delivery precision, and minimum DII result in a better performing system.

The BIUs and RMUs use two communication models: fixed delay and synchronous. In the **fixed-delay communication** model, the transmissions are triggered by events at the sources and the receiving nodes process the messages as soon as they arrive. For these transmissions, the information of interest is in the timing of the message. Fixed-delay communication is used by the synchronization protocols to enable receiving nodes to measure the relative skew of events at the source nodes. Likewise, this mode of communication enables the source nodes to estimate the time of some events at the receiving nodes. In the **synchronous communication** model, the transmissions are triggered at the local time indicated by the time-indexed operation schedule and the receiving nodes buffer the messages until their scheduled time for processing. This buffering of the messages implements a deskewing function that synchronizes the received messages to the local time at the receiving nodes. The relevant information carried by the messages is in their content. This model of communication is used by the synchronous protocols.

3.4. Distributed coordination

The functionality of the nodes and their interaction must be well specified in order to achieve the desired ROBUS behavior. The functional description of the mode logic, the diagnostic system, and the protocols specifies the required functions, their sequencing, and the nodes where the functions are to be performed. Those areas of the ROBUS design are discussed elsewhere in this document. This section examines timing aspects in the implementation of the protocols.

All the synchronization protocols are composed of the same basic operations. They are all event-driven, use only fixed-delay communication, and perform event voting using the Accept function. A requirement for all the synchronization protocols is that the Accept functions must receive all their corresponding valid inputs. However, the protocols differ substantially in the assumed initial precision. For this reason, the timing of their communication and computation processes is quite different. For Synchronization Preservation in the Clique Join and Clique Preservation modes, the nodes are assumed to be in the synchronized state with a relatively high synchronization precision. The beginning of this protocol is time triggered. Although the communication and computation processing are event-driven, it is possible to use protocol events and knowledge of their precision bound to determine local-time intervals for the expected times of transmission and reception. Based on this information, the Accept functions can be activated at specific local times when their corresponding input messages should be available. For Initial Synchronization in Clique Initialization mode, the nodes are in the unsynchronized state at the beginning of the protocol. The known bound of the starting synchronization precision enables the use of a time trigger for the protocol, but due to the large initial imprecision, no attempt is made to estimate the time of reception at the nodes executing the protocol. Because of this, the nodes must simply be ready to process the messages whenever they arrive. For Synchronization Acquisition in Clique Detection mode, the bus is assumed to be synchronized, but the recovering node is not. During this mode, a protocol is used to loosely synchronize to the re-synchronization interval of the clique operating in Clique Preservation mode. After this, the node must be ready to process the synchronization messages whenever they arrive. Appendix C analyzes the timing of the synchronization protocols in more detail, including the timing requirements for the internal processes of the nodes.

The synchronous protocols use time-triggered communication and processing. The ROBUS is able to execute synchronous protocols in the unsynchronized and synchronized states by exploiting the known bounds on the local-time synchronization precision in each state. The timing of execution of the synchronous protocols is specified by a time-indexed operation schedule. The scheduling of operations is based on a **distributed synchronous composition** abstract model of the system in which a single oscillator drives a common local-time clock and fixed-delay processes corresponding to the communication and computation operations of the BIUs and RMUs.

Figure 3.4 illustrates an example of the use of synchronous composition to schedule distributed processes. Element A of Figure 3.4 is the process dependency graph. P1 and P2 are computation processes, and COMM is a communication process. Element B of Figure 3.4 is the local time axis for the synchronous composition model. T_{P1}, T_{COMM}, and T_{P2} are the start times for the processes, and Δ_{P1}, Δ_{P2}, and Δ_{COMM} are the process delays. For proper operation, the schedule must satisfy the following constraints: $T_{COMM} \geq T_{P1} + \Delta_{P1}$, and $T_{P2} \geq T_{COMM} + \Delta_{COMM}$. For the actual system, processes P1 and P2 are performed on separate nodes driven by independent oscillators. Elements C and D of Figure 3.4 are the local time axes for the source and receiver nodes, respectively. The computation processes are simply started at their scheduled local times and run to completion using the computational resources of their respective nodes. The COMM process implements the synchronous communication. The message is sent (i.e., transferred to the link transmitter) by the source at the scheduled time. $\pi_{src\text{-}rcv}$ denotes the bound on

the synchronization precision, which corresponds to the uncertainty in the time of transmission measured in real time. The nominal link delay is denoted by Δ_{link}, and the link delay imprecision is e_{PP}. Considering these uncertainties, the expected time of reception is determined to be between T_L and T_H. If a message arrives earlier than T_H, it is buffered at least until T_H. Therefore, the communication delay used for scheduling purposes is $\Delta_{COMM} = T_H - T_{COMM}$. Appendix B presents a more detailed analysis of the point-to-point communication process. Note that the computation delays used for scheduling are the worst-case delays. Similarly to the communication process, if computation results are available earlier than expected, they must be buffered until the next communication process is started.

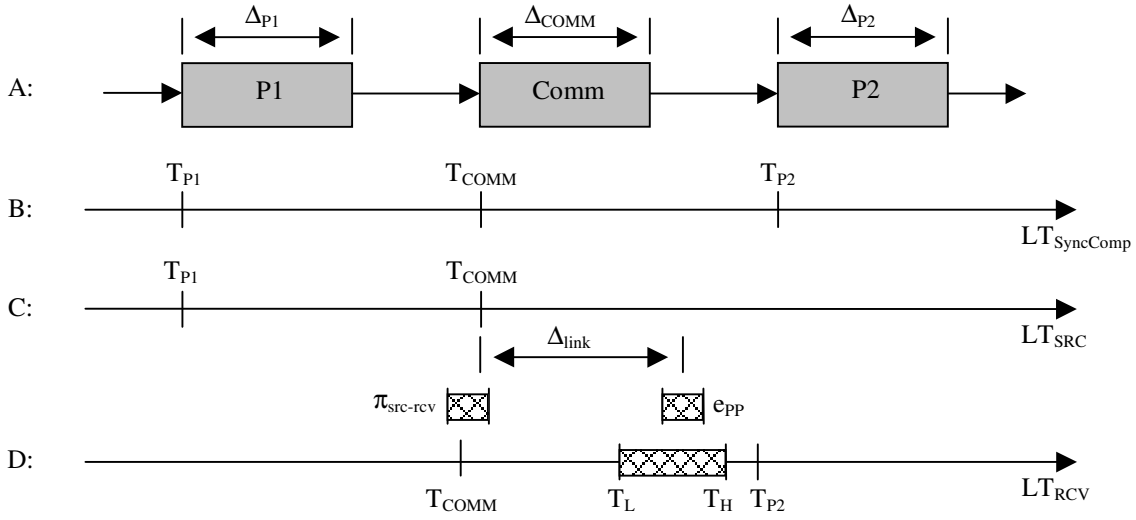


Figure 3.4: Coordinating distributed processes using synchronous composition

A: Dependency graph, B: Timeline for synchronous composition,

C: Timeline for source node, D: Timeline for receiver node

The synchronous composition model is applied to the scheduling of the protocols and their operations. The scheduling for an actual implementation must take into consideration the throughput capacity of the links, computation, and diagnostic processes, and the interactions between succeeding protocols, including the interactions between synchronous and synchronization protocols.

3.5. Communication between PEs and BIUs

The ROBUS requires a bidirectional communication capability between BIUs and their attached PEs. A BIU and its PE can be in separate FCRs or they can share an FCR. If a BIU and its PE are in separate FCRs, the physical communication links must provide adequate barriers to the propagation of faults between the FCRs. In this case, the physical failure of the BIU is independent from the failure of the PE and the failure recovery process of the BIU is completely independent from the PE. Note that if a BIU fails, its attached PE is in effect disconnected from the bus. On the other hand, if a BIU and its PE share an FCR, a fault can propagate between the BIU and the PE. In this case, the physical failure of one is no different from a failure of the other. Therefore, the design must provide for the simultaneous recovery of both components. For this version of the ROBUS, this is handled by a common process that resets the

BIU and the PE when a failure is detected on either of them. Irrespective of the FCR configuration, it is the responsibility of the PE to monitor the communication in order to determine the state of the BIU.

The BIUs and the PEs exchange messages using two communication models: fixed delay and synchronous. The fixed-delay model is used with the time synchronization protocols and is essentially the same as for the communication between BIUs and RMUs. The fixed delay allows the PEs to synchronize their time using events at the BIUs as references. Synchronous communication is used with the synchronous protocols. Two different synchronous communication models are allowed. In the "tight" model, the transmission of messages between BIUs and PEs follows a strict schedule in which timing is specified down to the tick level. This is the model used for synchronous communication between BIUs and RMUs. In the "loose" synchronous communication model, the sending and receiving of messages by the PEs is only required to satisfy simple timing constraints. Since the BIUs have time-triggered operation, their input and output of PE messages follows a detailed time-indexed schedule. The send timing requirement for the PEs is that their messages must be available at the BIUs at or before the time at which the BIUs will read them. The receive timing requirement for the PEs is that they will get the messages after the BIUs generate them according to their schedule. The specific time at which the PEs send their messages and the delay in receiving BIU messages for the "loose" synchronous communication model is dependent on the implementation and the applications run by the PEs. The PE Interface at the BIUs is designed using a first-in first-out (FIFO) buffer abstraction for input and output. For input, it is assumed that each expected PE message is available or there is a corresponding error indication. For output, it is always assumed that the message can be output at its scheduled time without having to confirm that the PE is ready to receive it.

The PEs send messages only during the schedule update and PE Broadcast services. In both cases, only DATA messages are sent. For these messages, the BIUs read the messages and broadcast them on the bus. PE-error checks are not described in this document. These checks are used to signal the BIU when expected messages are invalid or not available. In either case, a BIU will replace the expected PE message with a SPECIAL message with PE_ERROR payload field.

In addition to service messages, the PEs receive mode and identification messages from the BIUs. The mode messages enable the PEs to track the mode of their BIUs. A BIU will send a mode message to its PE every time there is a major mode transition and after every diagnostic cycle during the Clique Join and Clique Preservation modes. The mode messages are SPECIAL messages with the payload field set to SELF_TEST, CLIQUE_DETECTION, CLIQUE_INITIALIZATION, CLIQUE_JOIN or CLIQUE_PRESERVATION, as appropriate. The identification messages inform a PE of the identification number of its BIU, which is also the PE's identification number. These are DATA messages with the payload field equal to the BIU's identification number. This way of giving an identification number to a PE is preferred over setting it directly at the PE because it allows the use of generic software at the PEs and prevents a mismatch between the BIU and the PE identification numbers.

Figure 3.5 illustrates the message exchange between a BIU and its attached PE during the Clique Join and the Clique Preservation modes. The mode and identification messages are sent to the PE between the Self-Diagnosis and Schedule Update services. During Schedule Update, the BIU reads the schedule submitted by its PE and sends to the PE the results decided by the bus for each PE. This is followed by a single message indicating the assessment of the new schedule. This consists of a SPECIAL message with the payload set to VALID_SCHEDULE, ZERO_SCHEDULE (i.e., the schedule is valid and equal to zero for all the PEs), or INVALID_SCHEDULE. If the schedule is invalid, the ROBUS will automatically switch to a default schedule. During the PE Broadcast service, a BIU will read the scheduled messages from its PE and output to the PE the result for all the scheduled messages. A broadcast result equal to

PE_ERROR indicates that there was an error at the source PE. If the bus determines that the BIU of a source PE is not operating properly, then the result of the broadcast will be SOURCE_ERROR or NO_MAJORITY. A result of NO_MAJORITY indicates that the RMUs received different messages from the BIU. If the assessment of the schedule was ZERO_SCHEDULE, the PE Broadcast service is not executed and the ROBUS simply waits until it is time to execute the Time Reference service. During the Time Reference service, a BIU outputs a SPECIAL message with the payload set to INIT. The sending of this message is triggered by the reference event that the BIU will use to reset its local-time clock. For the Initial Synchronization and Synchronization Acquisition, the payload is set to ECHO to explicitly indicate that a different protocol event is used as a reference to reset the local-time clock. During the Self-Diagnosis service, the output of a BIU consists of two messages containing the diagnostic results for the BIUs and the RMUs. These are the last messages of the diagnostic cycle. The next messages are the mode and identification messages for the next cycle.

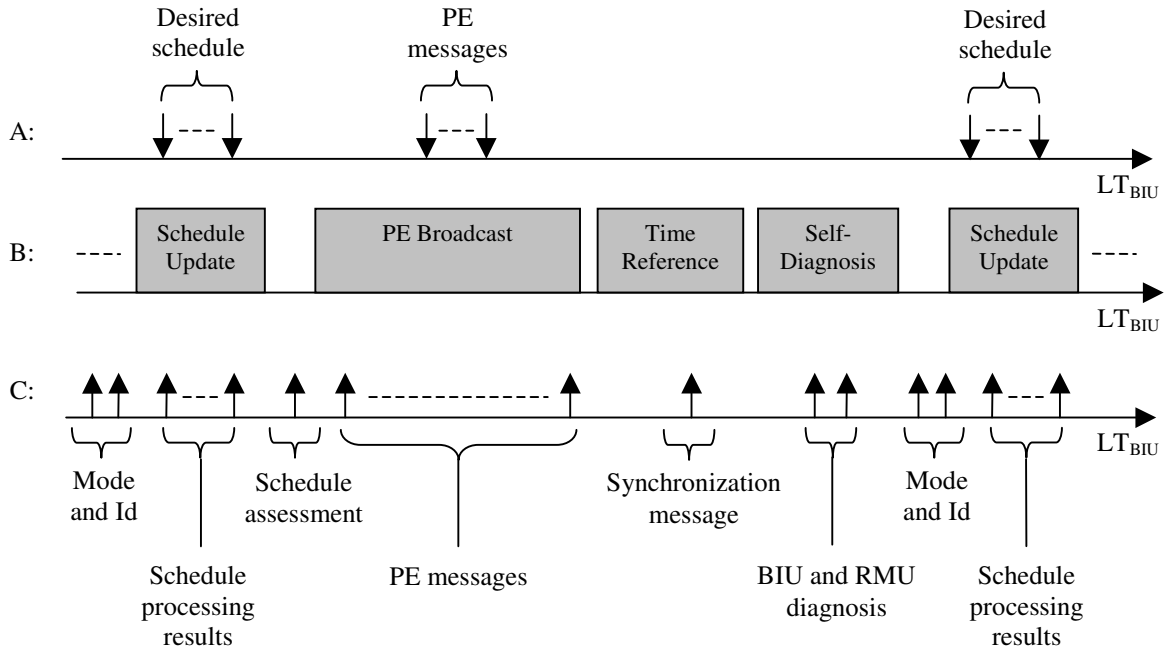


Figure 3.5: Message exchange pattern between a BIU and its attached PE in Clique Preservation mode

A: PE-to-BIU messages, B: Bus services, C: BIU-to-PE messages

4. Diagnostic system

Conceptually, the ROBUS nodes are composed of two separate but coordinated systems. The **operational system** handles the communication and computation activities required by the distributed protocols, in addition to all the processing associated with the mode logic, local time, and PE communication schedule. The **diagnostic system** monitors the operational system and provides timely information for reconfiguration and error containment. The main purpose of the diagnostic system is to ensure the continued operational survival of the bus. To support the fault-tolerance mechanisms, the diagnostic processes must work in close coordination with the operational system processes. The basic functions of the diagnostic system are to detect errors during the execution of the protocols, assess the status of individual nodes, and assess the status of the bus. The protocols use input-error detection and dynamic voting to protect against the propagation of errors into the error-containment regions (ECRs). Bus reconfiguration is based on the node assessment results generated by the diagnostic system. The ROBUS will continue to deliver services as long as the diagnostic system does not detect a failure of the bus.

The PEs can obtain diagnostic information from the bus in several ways. The ROBUS provides explicit periodic updates about the diagnostic status of every node. The protocol for the PE Broadcast service not only allows the broadcasting of messages, but it also diagnoses the BIUs as they transmit messages, and the results are forwarded to the PEs during the time of the service. In addition, the protocols for the PE Broadcast and Schedule Update services provide diagnostic information in the form of message content that indicates erroneous behavior by individual PEs or by the group of PEs attached to BIUs that are part of the active clique. Each individual PE can also use observations about the behavior of its attached BIU to derive additional information about the status of the BIU and the bus.

4.1. System structure

Each ROBUS node is an observer of every node. An observed node is known as a defendant. A direct observer receives information from the defendant by way of a direct data communication link. An indirect observer receives information from the defendant by way of direct observers. Due to the ROBUS topology, a node is a direct observer of nodes of its opposite kind and an indirect observer of nodes of its same kind, including itself. Every ROBUS node is a defendant and an observer.

Every ROBUS node performs the diagnostic functions of error detection, node assessment, and bus assessment. Error detection is the foundation of the diagnostic system. The **communication checks** monitor the communication links between the nodes. The **in-line checks** are applied to the received messages and are based on expected timing and content characteristics. The **cross-lane checks** also detect errors in received messages by comparing them against the result of dynamic voting. The **protocol checks** inspect received messages and voting results with respect to expected properties for intermediate and final protocol results. The **self-checks** are performed by a node to monitor its own operation. **PE-error checks** inspect the messages received by the BIUs from their attached PEs. All these error checks generate the syndromes from which diagnostic decisions are made.

The diagnostic system uses a distributed hierarchical classification system in which the severity of the diagnostic assessment for a defendant is related to the degree of certainty about its untrustworthiness. The diagnostic assessment of nodes is performed at the local and collective levels. Each ROBUS node gathers and processes diagnostic syndromes in order to form a local opinion about the status of each defendant, including itself. The nodes then share local assessment results to make a collective assessment

of each defendant. The overall assessment about the trustworthiness of a defendant is based on a combination of the local and collective assessments.

At the local level, each node uses knowledge about the protocols to interpret the generated error syndromes. A defendant is suspected by an observer when it determines that the defendant is one of several possible culprits for a detected error. In this case, the observer must combine multiple error syndromes to decide if a particular defendant is an error source. A defendant is blamed when an observer determines that the defendant is the source of a detected error. Blame can be assigned directly from the individual error syndromes or through the processing of suspicions. A defendant is accused when an observer independently determines that the defendant is untrustworthy but the observer is uncertain that other observers have reached the same conclusion. The accusations are based on assigned blame for detected errors.

At the collective level, the nodes exchange their accusations against defendants to form a common opinion about each one. The Collective Diagnosis protocol generates convictions by merging local diagnoses from direct and indirect observers for each defendant. A node is convicted when a sufficient number of observers consider the defendant untrustworthy.

The diagnostic assessment of the bus corresponds to assessing the status of the clique. The ROBUS nodes independently assess a clique based on locally available diagnostic information consisting of protocol check syndromes and the results of diagnostic node assessment.

The results of the diagnostic system are used for reconfiguration, error containment, and operational mode decisions. The membership of a clique (i.e., its configuration) is the set of nodes trusted to participate in collective operations. A clique is reconfigured by removing or adding nodes based on the results of diagnostic node assessment. Error containment is realized by input-error detection, dynamic voting, and fail-stop behavior. Input-error detection uses the communication and in-line checks. Dynamic voting requires the determination of inputs eligible to participate in a voting operation. Voting eligibility depends on the protocol and other specifics of the operation being executed. Fail-stop behavior falls under the categories of error containment and operational mode decisions. This behavior is triggered by the detection of critical conditions like a local failure or a clique failure. Other conditions relevant to operational mode decisions include the results of self-tests and the absence of a valid clique.

4.2. Diagnostic policy

The diagnostic policy specifies how the diagnostic data is to be processed to generate the required results. Among the factors taken into consideration in the design of the diagnostic policy are the following: the fault scenarios that the system is expected to encounter; the requirement to support the admission of new nodes into an existing clique; the requirement to support fault-tolerant startup and restart; and the requirement to ensure validity and agreement of diagnosis among clique members.

A clique is expected to survive, albeit in a degraded form, most scenarios of correlated transient faults in which external phenomena cause the simultaneous or nearly simultaneous failure of multiple nodes. To successfully handle these scenarios, the diagnostic system must rapidly reconfigure the clique by removing untrustworthy nodes before they have a chance to overwhelm the fault-masking mechanisms.

Recovering nodes must have a way to join a clique operating in Clique Preservation mode. This requires that the recovering nodes have access to the state of the clique, specially the local time and the

diagnostic state, and be able to acquire it.

The ROBUS design must provide mechanisms to allow trustworthy nodes to reach agreement on the local time and the membership of the clique when starting up or restarting the bus.

Most diagnostic decisions are independently made by the nodes based on their observations. The diagnostic system must satisfy certain general properties that ensure that the nodes are able to establish and maintain proper diagnostic-state agreement.

4.2.1. Required properties

A goal of the diagnostic system is to ensure that every untrustworthy defendant is eventually distrusted (i.e., **completeness**) and that only untrustworthy defendants are distrusted (i.e., **correctness**). If the fault model includes arbitrary asymmetric ("Byzantine") faults (see Appendix A), it is impossible to guarantee both of these properties [Shin 87]. The design of the ROBUS sacrifices completeness in order to ensure correctness.

The ROBUS diagnostic system must satisfy two basic properties: correctness and agreement for non-asymmetric defendants. The property of **correctness** requires that every distrusted defendant is indeed untrustworthy. Thus, for situations in which two or more defendants are possible culprits of a detected error, each must remain trusted unless there is additional evidence that indicates unambiguously that it is untrustworthy. This property ensures that the trusted set held by the observers includes all of the trustworthy defendants. The disadvantage of this required property is that the trusted set can also include untrustworthy ones. The main reason for requiring this property is to protect against premature exhaustion of available redundancy on the bus. The ROBUS includes a significant amount of error detection and diagnostic functions added to achieve high error coverage and increase the chances of successfully identifying untrustworthy nodes.

The property of **agreement for non-asymmetric defendants** requires that all of the trustworthy observers of a particular kind in a clique agree on their diagnostic assessment of defendants that are not asymmetric. To satisfy this property, all of the trustworthy members of a clique that are of the same kind must use the same mechanisms and common information to diagnose defendants. The only exception is when a node is diagnosing itself, in which case it is allowed to use exclusive local information.

4.2.2. General approach

The activities of the operational system are organized in a hierarchy with the following levels: **major mode**, **minor mode**, **protocol**, **(protocol) process**, and **(process) step**. The diagnostic system processing is directly dependent on the activity performed by the operational system. Every operational major mode has a corresponding diagnostic system mode involving two diagnostic intervals. The **local diagnostic interval** is time during which diagnostic information is gathered and processed locally in order to generate accusations. The **collective diagnostic interval** is the time between updates of the convictions. For this version of the ROBUS, the diagnostic information is represented in terms of Boolean variables (i.e., TRUE or FALSE). The **diagnostic state variables** of a node are the ones whose values are carried from one protocol process to the next. These include the suspicions, accusations, and convictions.

In the Clique Preservation mode, a clique tries to maintain validity and agreement on the time and diagnostic state variables of its members. Figure 4.1 shows the minor modes and the diagnostic intervals

for this major mode. The nodes gather local diagnostic information from the beginning of an execution of Collective Diagnosis to the beginning of the next. The locally generated accusations are submitted for processing at the beginning of Collective Diagnosis and a consistent update to the convictions is received at the end. At the beginning of Collective Diagnosis, the nodes copy their current local accusations to a temporary memory location and then clear (i.e., set to FALSE) their suspicions and accusations to start the next local diagnostic interval. During Collective Diagnosis, the effective accusations are formed by a bitwise OR function of the accusations stored in the temporary memory location and any newly generated accusations. The accusations in temporary memory are discarded at the end of Collective Diagnosis and only accusations from the current local diagnostic interval are used from that point on. The overlap in the use of accusations from consecutive local diagnostic intervals ensures that the clique remains guarded from untrustworthy nodes during Collective Diagnosis and that the local accusations processed during Collective Diagnosis are based on observations gathered during the full local diagnostic interval.

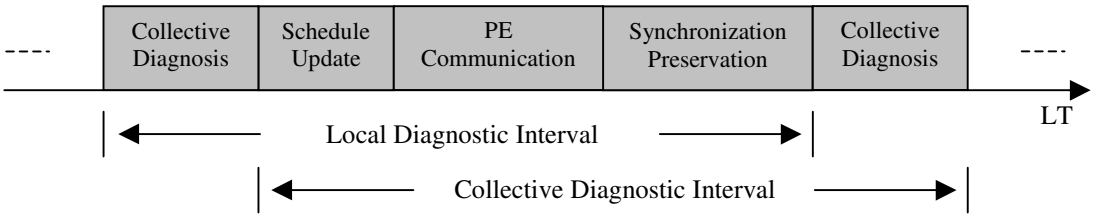


Figure 4.1: Diagnostic intervals for the Clique Preservation mode

The processing done by the diagnostic system during the Clique Join mode is essentially the same as for the Clique Preservation mode. The only difference is that a node in the Clique Join mode expects not to be a member of the clique when it enters this mode and to be part of it at the end.

In the Clique Detection mode, a node tries to determine if a clique is operating on the bus. This is accomplished by operating as if a clique were present and attempting to acquire its time and diagnostic state, while simultaneously monitoring for indications that a valid clique is not present. It is assumed that at any time there is at most one clique present on the bus. The error processing by a node in this mode is based on the assumption that the node is working properly unless there is unequivocal evidence that it is not. In the absence of such evidence, all detected errors involving other nodes or the clique are blamed on them. Figure 4.2 shows the minor modes and the range of the diagnostic intervals for the Clique Detection mode. A node clears all its diagnostic state variables at the beginning of this mode, and the convictions are held constant until an update is received during Collective Diagnosis Acquisition. Only local diagnostic information is used to assess the nodes and the clique. During Local Diagnosis Acquisition, a node allows its diagnostic system to perform a preliminary assessment of the bus before the operational system attempts to synchronize to the clique during the Synchronization Acquisition mode. During the Collective Diagnosis mode, a node loads the convictions computed by the clique. The operation of the diagnostic system during Collective Diagnosis is the same as described previously for nodes in Clique Preservation mode.

In the Clique Initialization mode, a group of nodes tries to form a new clique after failing to find a valid clique in the Clique Detection mode. Figure 4.3 shows the minor modes and diagnostic cycles. None of the diagnostic information gathered during Clique Detection is used in this mode. All of the diagnostic state variables are cleared at the beginning of this mode, and the convictions are held constant until an update is computed. As in Clique Detection mode, in this mode only local diagnostic information is used to assess the nodes and the clique. During Initial Diagnosis, the nodes determine the initial set of

nodes that will be considered to form the new clique. These nodes agree on the local time during the Initial Synchronization mode, and new convictions are computed during Collective Diagnosis. The operation of the diagnostic system during Collective Diagnosis is the same as described previously for nodes in Clique Preservation mode.

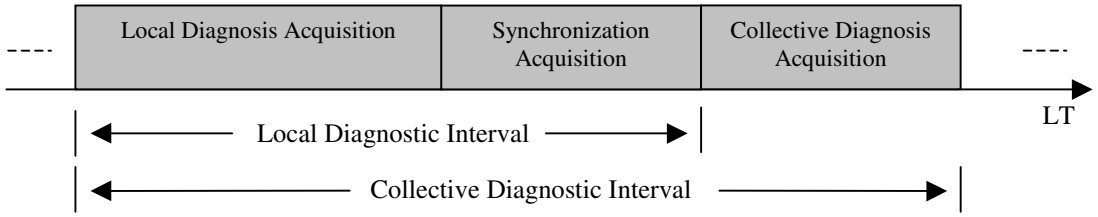


Figure 4.2: Diagnostic intervals for the Clique Detection mode

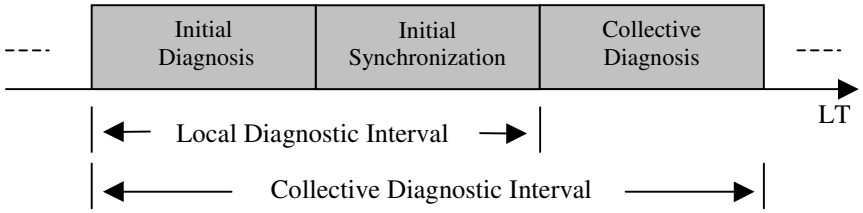


Figure 4.3: Diagnostic intervals for the Clique Initialization mode

In the Self-Test mode, the operational system exercises the circuitry of a node while the diagnostic system monitors for errors. No diagnostic information about other nodes is gathered during this mode.

4.2.3. Suspicion generation

For some operational scenarios, an observer is able to detect that an error has occurred in a particular communication path that starts at a source node of the same kind as the observer, passes through an intermediate node of the opposite kind, and ends at the observer. The observer always assumes that its own operation is correct unless there is evidence to the contrary. In the absence of other error syndromes directly incriminating the source node or the intermediate node, the observer cannot determine which is responsible for the detected error. Therefore, the best that the observer can do is to raise suspicions against both of them. For this version of the ROBUS, suspicions are generated only for such scenarios.

Figure 4.4 illustrates the organization of suspicions in a two-dimensional matrix in which the rows correspond to the nodes of the same kind as the observer and the columns correspond to the nodes of the opposite kind. Θ and Ω denote the number of nodes of the same kind and of the opposite kind, respectively. Every $S_{i,j}$ cell is a Boolean variable. A single instance of suspicion against a pair of nodes is sufficient to assert (i.e., set to TRUE) the corresponding cell in the matrix.

		Opposite Kind			
		1	2	…	Ω
Same Kind	1	$S_{1,1}$	$S_{1,2}$	…	$S_{1,\Omega}$
	2	$S_{2,1}$	$S_{2,2}$	…	$S_{2,\Omega}$
	…	…	…	…	…
	Θ	$S_{\Theta,1}$	$S_{\Theta,2}$	…	$S_{\Theta,\Omega}$

Figure 4.4: Structure of the suspicions variable

4.2.4. Accusation generation

A node has accusations variables for every node on the bus. Figure 4.5 illustrates the organization of the accusations variables. Every $A_{SK,i}$ or $A_{OK,j}$ cell is a Boolean variable that is asserted when the corresponding node is accused. An observer accuses a defendant when it determines that the defendant is responsible for one or more detected errors.

Same Kind				Opposite Kind			
1	2	…	Θ	1	2	…	Ω
$A_{SK,1}$	$A_{SK,2}$	…	$A_{SK,\Theta}$	$A_{OK,1}$	$A_{OK,2}$	…	$A_{OK,\Omega}$

Figure 4.5: Accusations variables

There are two accusation generation mechanisms. For most error checks, only one defendant can be blamed and a single detected error is sufficient evidence to accuse the defendant. The other accusation generation mechanism is the processing of the suspicions matrix. This generates accusations using bit vote operations for each row and each column of the suspicions matrix. Only rows and columns corresponding to trusted nodes are considered. Figure 4.6 illustrates this for a system with 4 nodes of the same kind and 3 nodes of the opposite kind. $A_{OK,i}|_{susp}$ denotes the suspicions-based accusation result for the i-th node of the opposite kind. These accusations are generated by bit voting considering only the suspicions in the rows corresponding to trusted nodes of the same kind. Suspicions-based accusations against nodes of the same kind are similarly determined. A bitwise Boolean OR operation for each defendant is used to combine the corresponding results of the accusation generation mechanisms.

The accusations variables remain constant during the execution of protocol processes and are updated after the completion of the protocol process in which the incriminating evidence is found. The suspicion matrix is processed only at the end of the PE Communication mode. The accusations and suspicions are cleared at the end of the local diagnostic interval.

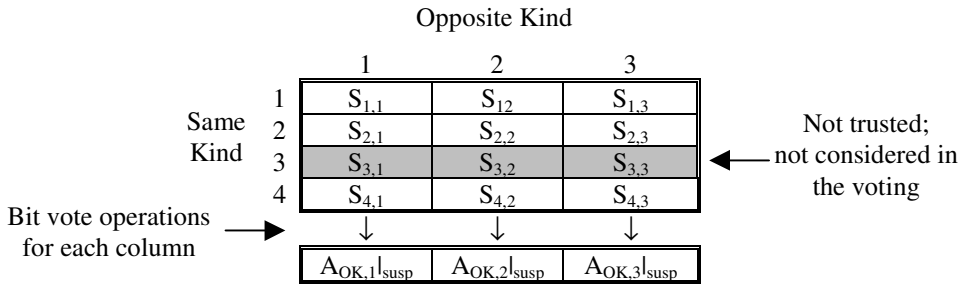


Figure 4.6: Example of the generation of suspicions-based accusations for nodes of the opposite kind

4.2.5. Conviction generation

Figure 4.7 illustrates the organization of the convictions variables. Every $C_{SK,i}$ or $C_{OK,j}$ cell is a Boolean variable that is asserted when the corresponding node is convicted. The convictions are generated by the Collective Diagnosis protocol. Bit vote operations for accusations from eligible voters are used to determine the conviction result for each defendant. The Collective Diagnosis protocol is presented in Section 5. The conviction variables remain constant until updated at the end of the collective diagnostic interval.

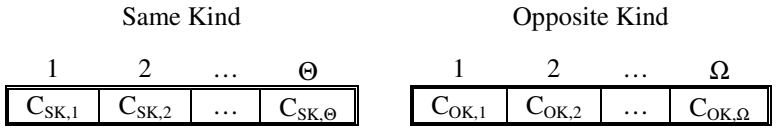


Figure 4.7: Convictions variables

4.2.6. Trust

A node uses the results of local and collective diagnoses to determine which nodes to trust. The general rule to determine trust is that a node is trusted if it is not accused and not convicted. This rule applies to all operational modes. Both of these diagnosis results are available to nodes in the Clique Join and Clique Preservation modes. During the Clique Detection and Clique Initialization modes, only local diagnostic information is available until the end of the collective diagnostic cycle. Since a node must be trusted unless there is evidence that it is untrustworthy, nodes operating in these modes must clear their convictions variables and, in effect, use only the accusations to determine trust.

4.2.7. Voter eligibility

Dynamic voting is applied to received messages and gathered suspicions. Distrusted nodes are not allowed to participate in voting operations. In addition, detected input errors and message content may be considered to determine voter eligibility for received messages. The eligibility conditions are independently determined for each instance of voting.

4.2.8. Local failure and bus failure conditions

The trigger for a node to stop its current activity and transition to the Self-Test mode is the detection of a local failure or a bus failure. The status of the bus is always determined in terms of the status of the clique. The application of error checks and the interpretation of their syndromes are dependent on the activities performed by the operational system. For most checks, a detected error can be caused by a failure of the observer or the observed object, which is either a particular defendant or a clique as a whole. The assessment of individual nodes is always performed with the assumption that the observer and the clique are working properly. In general, an observer can assess its own status by using clique-dependent checks for which a clique is assumed to be operating properly, or by using clique-independent checks that monitor the observer for conformance with its design specification. The latter type of self-checks is an optional feature whose use should be decided with consideration given to factors like implementation complexity and the application domain. All of the self-checks described in this document are clique-dependent checks. Thus, there is no completely unambiguous way for a node to differentiate between a local failure and a clique failure.

In the Clique Initialization, Clique Join, and Clique Preservation modes there is no need to distinguish between a local failure and a clique failure because the detection of either always triggers a fail-stop response.

In the Self-Test mode, only local operations are performed and no information from the rest of the bus is processed. In this mode, any detected error is presumed to be due to a local failure.

In the Clique Detection mode, three events trigger abrupt transitions: local failure, clique failure, and no clique found. In this mode, no distinction is made between the case in which a clique is not present when the observer transitions to this mode and the case in which a clique experiences a failure while the observer is in this mode. Both cases are considered indications that a clique is not present. In addition, a node in this mode assesses a clique based on the assumption that the node itself is operating properly unless there is an unambiguous indication that it is not. Given that the operations defined for this mode do not allow a node to observe its own behavior, the only such indication is an accusation against itself. This is discussed in detail in Section 7.

A clique is diagnosed based on its size and the validity and agreement of its processing results. The smallest clique allowed is composed of one BIU and one RMU. To assess a clique for size, no distinction is made between BIUs or between RMUs. A node uses its trusted set and the results of Collective Diagnosis to determine the membership of a clique. Most protocol checks monitor a clique for validity and agreement in state variables and processing results. For some input voting operations, an observer expects agreement among a majority of the eligible voters, or it knows in advance what the result of the voting should be.

4.2.9. Unexpected messages

For most modes of operation, a node expects to receive messages during particular local-time intervals. This includes all of the synchronous protocols and the Synchronization Preservation protocol. The reception of a message when none is expected is an indication of a timing error. In agreement with the accusation generation policy, the detection of such an error should result in an accusation against the corresponding node of the opposite kind.

In order to ensure proper coordinated action among the trustworthy members of a clique, the design of the ROBUS nodes must ensure that untrustworthy nodes do not have a chance to influence the time at which individual trustworthy nodes update their accusations. This is realized by updating the accusations only at predetermined points in time. For this version of the ROBUS, this is done at the beginning and at the end of protocol processes. Thus, if an unexpected message is detected in between protocol processes, the corresponding accusation becomes effective at the start of the next process in which input messages are expected. If the error detection occurs during a process, the accusation becomes effective after the completion of the process.

5. Clique Preservation

Figure 5.1 illustrates the minor mode transitions for the Clique Preservation major mode. A node enters this mode after confirming that it has been admitted to the clique at the end of a diagnostic cycle in Clique Initialization or Clique Join modes. In the Clique Preservation mode, a node participates in the delivery of services to the PEs in a continuous loop. Each one of these services is realized with a specialized distributed protocol. The only exit condition for the Clique Preservation mode is the detection of a local failure or a bus failure.

The local failure or bus failure conditions are either protocol-independent or protocol-dependent. The protocol-independent conditions include each of the following: number of trusted BIUs equal to zero, number of trusted RMUs equal to zero, and assertion of an accusation against self. The protocol-dependent conditions are described with the corresponding protocols.

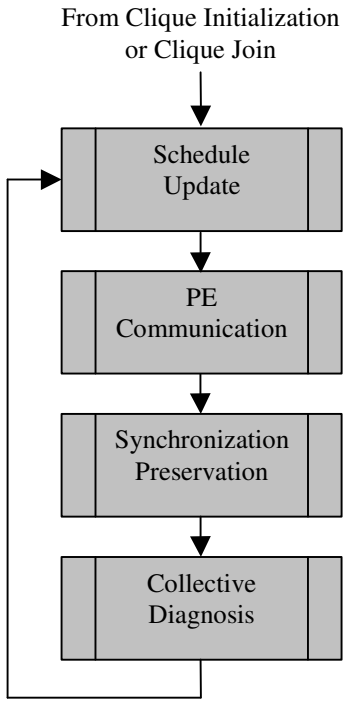


Figure 5.1: Minor-mode transitions for Clique Preservation mode

5.1. Schedule Update

In the Schedule Update mode, the PEs submit their desired schedule for the next PE Broadcast communication service. Ideally, the PEs have agreement on the schedule before they deliver it to the ROBUS. Let N denote the number of BIUs, which is assumed to equal the number of PEs connected to the bus. The desired schedule is delivered by each PE to its BIU in the form of N consecutive messages with the positions in the sequence corresponding to the identification numbers of the PEs and the payload fields of the messages indicating the desired number of messages to be broadcast. The submitted schedule messages are processed using an agreement protocol to ensure that all of the clique members and the PEs agree on the result for each PE. The protocol is applied independently N times, with each

iteration processing the messages delivered by the PEs that indicate the number of messages to be broadcast by a particular PE. After all the messages have been processed, the ROBUS nodes individually assess the resulting schedule. Since there is agreement on the protocol results and the nodes apply the same assessment rules, their assessment results are guaranteed to be the same.

Note that this protocol is executed by nodes in the Clique Preservation and Clique Join modes.

5.1.1. Schedule Update protocol

The Schedule Update protocol was developed for ROBUS based on the theory presented in [Miner 04]. Figure 5.2 shows the message flow graph. The labels inside the circles identify the processes executed by the ROBUS nodes. This protocol is a synchronous protocol implemented using synchronous communication.

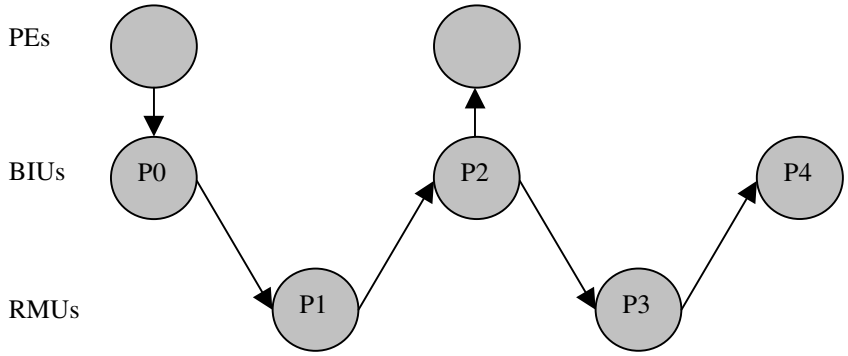


Figure 5.2: Message flow graph for the Schedule Update protocol

The following is the description of the basic Schedule Update protocol. This protocol determines the number of messages to be broadcast by the i-th PE according to the schedule messages submitted to the ROBUS by the PEs. The sublevels in the description specify the checks to be performed during particular protocol steps. All of the checks with specific expectations will result in errors being signaled if the expectations are not met. Note that the ROBUS messages are expressed in functional notation: RM(tag, payload). A message with an arbitrary payload field is indicated by a '*' symbol in the payload location.

Process P0: BIUs
1. If the PE message is valid, broadcast that message. Otherwise, broadcast RM(SPECIAL, PE_ERROR).

Process P1: RMUs
1. Receive the messages from the BIUs.
  1.1. Communication checks for each BIU:
    1.1.1. Expecting no link errors

1.2. In-line checks for each BIU:
1.2.1. Expecting reception within a predetermined local-time interval
1.2.2. Expecting exactly one message
1.2.3. Expected content: RM(SPECIAL, PE_ERROR), RM(DATA, *)

2. For each BIU, if there was a reception error, the message content is ineligible, or the BUI is not trusted, then the BIU is not an eligible voter.
2.1. Eligible content for each BIU: RM(DATA, *)

3. Perform a word vote on the messages from eligible voters. If there is no majority or there are no eligible voters, then convert the result to RM(SPECIAL, PE_ERROR).

4. Broadcast the result of the vote.

Process P2: BIUs

1. Receive the messages from the RMUs.
1.1. Communication checks for each RMU:
1.1.1. Expecting no link errors
1.2. In-line checks for each RMU:
1.2.1. Expecting reception within a predetermined local-time interval
1.2.2. Expecting exactly one message
1.2.3. Expected content: RM(SPECIAL, PE_ERROR), RM(DATA, *)

2. For each RMU, if there was a reception error, the message content is ineligible, or the RMU is not trusted, then the RMU is not an eligible voter.
2.1. Eligible content for each RMU: RM(SPECIAL, PE_ERROR), RM(DATA, *)
2.2. Protocol checks:
2.2.1. Expecting that at least one RMU is an eligible voter

3. Perform a word vote on the messages from eligible voters. If there is no majority, then convert the result to RM(SPECIAL, PE_ERROR).

4. Broadcast the result.

5. Send the result of the vote to the attached PE.

Process P3: RMUs

1. Receive the messages from the BIUs.
1.1. Communication checks for each BIU:
1.1.1. Expecting no link errors
1.2. In-line checks for each BIU:
1.2.1. Expecting reception within a predetermined local-time interval
1.2.2. Expecting exactly one message
1.2.3. Expected content: RM(SPECIAL, PE_ERROR), RM(DATA, *)

2. For each BIU, if there was a reception error, the message content is ineligible, or the BUI is not trusted, then the BIU is not an eligible voter.
  2.1. Eligible content for each BIU: RM(SPECIAL, PE_ERROR), RM(DATA, *)
  2.2. Protocol checks:
    2.2.1. Expecting that at least one BIU is an eligible voter

3. Perform a word vote on the messages from eligible voters.
  3.1. Cross-lane checks for each BIU:
    3.1.1. Expecting agreement with the result of the vote
  3.2. Protocol checks:
    3.2.1. Expecting agreement among a majority of the eligible voters

4. Broadcast the result of the vote.

5. The result of the vote is the protocol result for the i-th PE.

Process P4: BIUs

1. Receive the messages from the RMUs.
  1.1. Communication checks for each RMU:
    1.1.1. Expecting no link errors
  1.2. In-line checks for each RMU:
    1.2.1. Expecting reception within a predetermined local-time interval
    1.2.2. Expecting exactly one message
    1.2.3. Expected content: RM(SPECIAL, PE_ERROR), RM(DATA, *)

2. For each RMU, if there was a reception error, the message content is ineligible, or the RMU is not trusted, then the RMU is not an eligible voter.
  2.1. Eligible content for each RMU: RM(SPECIAL, PE_ERROR), RM(DATA, *)
  2.2. Protocol checks:
    2.2.1. Expecting that at least one RMU is an eligible voter

3. Perform a word vote on the messages from eligible voters.
  3.1. Cross-lane checks for each RMU:
    3.1.1. Expecting agreement with the result of the vote
  3.2. Protocol checks:
    3.2.1. Expecting agreement among a majority of the eligible voters

4. The result of the vote is the protocol result for the i-th PE.

A reception error, also called an input error, is a violation of the expectations for the communication or in-line checks. An error detection by any of these checks in processes P1 through P4 is sufficient evidence to accuse the corresponding node of the opposite kind

Note that in process P1 the expected content and the eligible content are not the same. The content RM(SPECIAL, PE_ERROR) is a valid input, but it is not eligible to determine the voting result.

The cross-lane checks in processes P3 and P4 compare the result of the vote with the received input from each node of the opposite kind. An error detection by these checks is sufficient evidence to accuse the corresponding node of the opposite kind.

Two types of protocol checks are used. In processes P2 through P4, it is expected that at least one node of the opposite kind is eligible to vote. In processes P3 and P4, it is expected to have agreement on received message content for a majority of the eligible voters. An error detection by any of these checks is an indication of a clique failure.

5.1.2. Schedule update assessment

The Schedule Update protocol determines the number of messages to be broadcast by a particular PE. The result of the protocol can be a DATA message with the number of scheduled messages in the payload field, or a PE_ERROR message indicating that the protocol was unable to determine a valid number of messages for the PE being considered.

The assessment of the schedule update produces one of three results: invalid, valid, or zero. A schedule is **invalid** if the result of the Schedule Update protocol is PE_ERROR for any PE, or if the total number of scheduled messages exceeds the maximum number of messages that the ROBUS can process during the PE Communication mode. This maximum number is a constant during run-time and is determined by doing a timing analysis of the bus implementation. A schedule is **valid** if it is not invalid. A **zero** schedule is a special case of a valid schedule in which the number of scheduled message is zero for every PE.

After completing their assessment, the BIUs send a SPECIAL message to the PEs to inform them of the result. The payload field is one of the following: INVALID_SCHEDULE, VALID_SCHEDULE, or ZERO_SCHEDULE.

5.1.3. Application of the schedule update assessment

The schedule for the next PE broadcast session depends on the result of the schedule update assessment. If the result is valid, the new schedule is used. If the result is zero, there will be no bus activity during the next PE Communication mode. If the result is invalid, a default schedule is used. The default schedule is constant during run-time and is known to all the ROBUS nodes and the PEs. For this version of the ROBUS, the default schedule allocates the same number of transmissions for each PE.

5.2. PE Communication

In the PE Communication mode, the ROBUS broadcasts PE messages according to the communication schedule. The access pattern is a time-indexed round-robin sequence in which the interval between the send times of consecutive messages is constant regardless of whether they are from the same or different PEs. The BIUs (and their attached PEs) access the bus in ascending order according to their identification numbers. If a particular PE is not scheduled to send messages, the next one that is scheduled will automatically take its place to ensure that the proper interval between messages is maintained. Each PE message is broadcast using an agreement protocol to ensure that the PEs receive the same result.

After all the scheduled messages have been processed, the BIUs and RMUs exchange their accumulated accusations against nodes of the opposite kind. This exchange enhances the reconfiguration capabilities of the bus by ensuring that the required diagnosis properties are satisfied when processing the accumulated suspicions. This is explained in detail in Appendix F.

The suspicions matrix is processed following the completion of the Accusation Exchange protocol. Section 4 of this document describes this operation.

Note that these protocols are executed by nodes in the Clique Preservation and Clique Join modes.

5.2.1. PE Broadcast protocol

The PE Broadcast protocol was inspired by the source congruency protocol presented in [Smith 84] for Drapper Lab's Fault Tolerant Processor (FTP) architecture. Figure 5.3 illustrates the message flow graph. This protocol is a synchronous protocol implemented using synchronous communication. The scheduled message is generated by the source PE and relayed by the attached BIU (a.k.a. the source BIU). The result of the protocol is received by all the PE attached to BIUs that are part of the clique or are in Clique Join mode.

The protocol is an agreement protocol with embedded diagnostic processing. If the source PE sends a valid message and its attached BIU is working properly, then the PEs will receive the message sent. A result of PE_ERROR indicates that the source BIU did not receive a valid message from the PE. SOURCE_ERROR indicates that there was an error caused by the source BIU. A result of NO_MAJORITY means that the source BIU did not broadcast the same message to each of the RMUs.

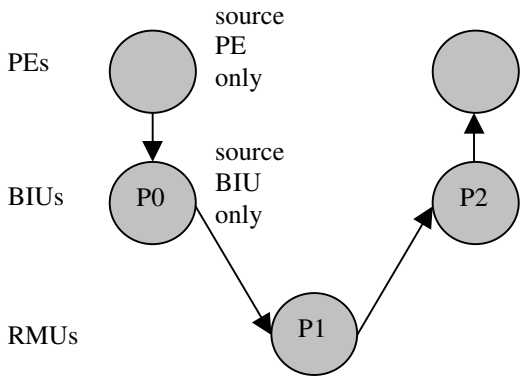


Figure 5.3: Message flow graph for the PE Broadcast protocol

The PE Broadcast protocol is presented next.

Process P0: Source BIU
1. If the PE message is valid, broadcast that message. Otherwise, broadcast RM(SPECIAL, PE_ERROR).

Process P1: RMUs

1. Receive the message from the source BIU.
   1.1. Communication checks for the source BIU:
      1.1.1. Expecting no link errors
   1.2. In-line checks for the source BIU:
      1.2.1. Expecting reception within a predetermined local-time interval
      1.2.2. Expecting exactly one message
      1.2.3. Expected content: RM(SPECIAL, PE_ERROR), RM(DATA, *)

2. For the source BIU, if there was a reception error, the message content is ineligible, or there is an accusation against it, then the BIU is ineligible.
   2.1. Eligible content for the source BIU: RM(SPECIAL, PE_ERROR), RM(DATA, *)

3. If the source BIU is eligible, the result is the received message. Otherwise, the result is RM(SPECIAL, SOURCE_ERROR).

4. Broadcast the result.

Process P2: BIUs

1. Receive the messages from the RMUs.
   1.1. Communication checks for each RMU:
      1.1.1. Expecting no link errors
   1.2. In-line checks for each RMU:
      1.2.1. Expecting reception within a predetermined local-time interval
      1.2.2. Expecting exactly one message
      1.2.3. Expected content: RM(SPECIAL, PE_ERROR), RM(SPECIAL, SOURCE_ERROR), RM(DATA, *)

2. For each RMU, if there was a reception error, the message content is ineligible, or the RMU is not trusted, then the RMU is not an eligible voter.
   2.1. Eligible content for each RMU: RM(SPECIAL, PE_ERROR), RM(SPECIAL, SOURCE_ERROR), RM(DATA, *)
   2.2. Protocol checks:
      2.2.1. Expecting that at least one RMU is an eligible voter

3. Perform a word vote on the messages from eligible voters. If there is no majority, then convert the result to RM(SPECIAL, NO_MAJORITY).
   3.1. Cross-lane checks for each RMU:
      3.1.1. Expecting agreement with the result of the vote
   3.2. Protocol check:
      3.2.1. Expecting that the vote result is not equal to RM(SPECIAL, NO_MAJORITY) or RM(SPECIAL, SOURCE_ERROR)
   3.3. Self-check: (source BIU only)
      3.3.1. Is the vote result equal to the message broadcast in process P0?

4. If the source is not trusted, then convert the result to RM(SPECIAL, SOURCE_ERROR).

5. Send the result to the attached PE.

In processes P1 and P2, a reception error is sufficient evidence to accuse the corresponding node of the opposite kind.

In processes P2, it is expected that at least one node of the opposite kind is eligible to vote. An error detection by this check is an indication of a clique failure. A vote result of RM(SPECIAL, SOURCE_ERROR) or RM(SPECIAL, NO_MAJORITY) in process P2 is an indication of an error by the source BIU and is sufficient evidence to accuse it. If the vote result in process P2 is not RM(SPECIAL, SOURCE_ERROR) or RM(SPECIAL, NO_MAJORITY), and an error is detected for a cross-lane check, then it is known that one or both of the corresponding RMU and the source BIU is responsible for the error. This is sufficient evidence to generate a suspicion against the RMU and the source BIU.

The self-check in process P2 is stated as a question because the expected result depends on the mode of the node executing this protocol process. For a node in the Clique Preservation mode or the Clique Join mode with its output enabled, it is expected that the result of the vote is equal to the message broadcast in process P0. For a node in the Clique Join mode with its output disabled, it is expected that the message broadcast in process P0 and the voting result in process P2 do not match. (Furthermore, the voting result should be SOURCE_ERROR for a node in the Clique Join mode with its output disabled.) Irrespective of the operating mode, a violation of the expectation for this check is an indication of a local failure.

5.2.2. Accusation Exchange protocol

Figure 5.4 shows the message flow graph for the Accusation Exchange protocol. Only BIUs and RMUs participate in this protocol. This protocol is a synchronous protocol implemented using synchronous communication.

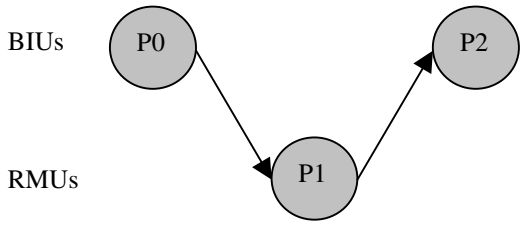


Figure 5.4: Message flow graph for the Accusation Exchange protocol

The description of the protocol is presented next.

Process P0: BIUs
1. Broadcast the local accusations against the RMUs.

Process P1: RMUs
1. Receive the messages from the BIUs.
  1.1. Communication checks for each BIU:
    1.1.1. Expecting no link errors
  1.2. In-line checks for each BIU:
    1.2.1. Expecting reception within a predetermined local-time interval
    1.2.2. Expecting exactly one message
    1.2.3. Expected content: RM(DATA, *)

2. For each BIU, if there was a reception error, the message content is ineligible, or the BUI is not trusted, then the BIU is not an eligible voter.
  2.1. Eligible content for each BIU: RM(DATA, *)
  2.2. Protocol checks:
    2.2.1. Expecting that at least one BIU is an eligible voter

3. For each RMU defendant, perform a bit vote on the accusations received from eligible voters.
  3.1. Cross-lane checks for each BIU:
    3.1.1. Expecting agreement with the result of the vote for each RMU defendant

4. Broadcast the local accusations against the BIUs.

5. For each RMU defendant, merge the result of the bit vote with the local accusation value. The result is the new local accusation value for the defendant.

Process P2: BIUs
1. Receive the messages from the RMUs.
  1.1. Communication checks for each BIU:
    1.1.1. Expecting no link errors
  1.2. In-line checks for each BIU:
    1.2.1. Expecting reception within a predetermined local-time interval
    1.2.2. Expecting exactly one message
    1.2.3. Expected content: RM(DATA, *)

2. For each RMU, if there was a reception error, the message content is ineligible, or the RMU is not trusted, then the RMU is not an eligible voter.
  2.1. Eligible content for each RMU: RM(DATA, *)
  2.2. Protocol checks:
    2.2.1. Expecting that at least one BIU is an eligible voter

3. For each BIU defendant, perform a bit vote on the accusations received from eligible voters.
  3.1. Cross-lane checks for each RMU:
    3.1.1. Expecting agreement with the result of the vote for each BIU defendant

4. For each BIU defendant, merge the result of the bit vote with the local accusation value. The result is the new local accusation value for the defendant.

Bitwise OR functions implement the merge operations in processes P1 and P2 between the results of the bit votes and the local accusations. The updates to the accusations become effective after the corresponding protocol process is complete.

In processes P1 and P2, a reception error is sufficient evidence to accuse the corresponding node of the opposite kind.

In processes P1 and P2, it is expected that at least one node of the opposite kind is eligible to vote. An error detection by this check is an indication of a clique failure.

For each bit vote operation in processes P1 and P2, if an error is detected for a cross-lane check, then it is known that the defendant for the bit vote, the node of the opposite kind for which the error was detected, or both are untrustworthy. This is sufficient evidence to generate a suspicion against the defendant and the node of the opposite kind.

5.3. Synchronization Preservation

In the Synchronization Preservation mode, the ROBUS executes a distributed synchronization protocol to re-synchronize the local-time clocks of the BIUs and RMUs, and to provide the PEs with a common time reference. It is assumed that the clique is already synchronized within some known precision bound. The protocol is intended to improve the precision by eliminating any relative skew introduced by the drift rate of the oscillators since the last synchronization.

The Synchronization Preservation protocol was originally inspired by the clock synchronization protocol presented in [Srikanth 87], and it is an extension of the synchronization protocol in [Miner 02]. Figure 5.5 shows the message flow graph for the protocol. This protocol is a synchronization protocol implemented using fixed-delay communication. The labels next to the arrows indicate the type of SPECIAL message that is transmitted by the sending process. The protocol is an agreement generation protocol with provisions for nodes trying to synchronize to the clique.

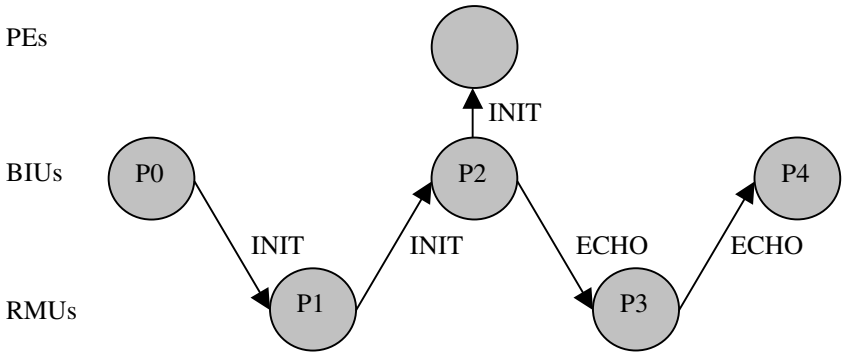


Figure 5.5: Message flow graph for the Synchronization Preservation protocol

The description of the protocol is presented next. Agreement checks are performed by comparing the time between relevant events against an expected duration. A synchronization-reset timer implements a delay between a reference event and the resetting of the local-time clock. Appendix C provides additional information about the timing aspects of this protocol.

Process P0: BIUs
1. If it is time to send, broadcast RM(SPECIAL, INIT).

Process P1: RMUs
1. Receive the messages from the BIUs.
  1.1. Communication checks for each BIU:
    1.1.1. Expecting no link errors
  1.2. In-line checks for each BIU:
    1.2.1. Expecting reception within a predetermined local-time interval
    1.2.2. Expecting exactly one message
    1.2.3. Expected content: RM(SPECIAL, INIT)

2. For each BIU, if there was a reception error, the message content is ineligible, or the BUI is not trusted, then the BIU is not an eligible voter.
  2.1. Eligible content for each BIU: RM(SPECIAL, INIT)
  2.2. Protocol checks:
    2.2.1. Expecting that at least one BIU is an eligible voter

3. Compute the Accept function for the messages received from the eligible voters.
  3.1. Cross-lane checks for each BIU:
    3.1.1. Expecting agreement with the result of the Accept function
  3.2. Protocol checks:
    3.2.1. Expecting agreement among a majority of the eligible voters

4. When the Accept output is asserted, broadcast RM(SPECIAL, INIT).

Process P2: BIUs

1. Receive the messages from the RMUs.
   1.1. Communication checks for each RMU:
      1.1.1. Expecting no link errors
   1.2. In-line checks for each RMU:
      1.2.1. Expecting reception within a predetermined local-time interval
      1.2.2. Expecting exactly one message
      1.2.3. Expected content: RM(SPECIAL, INIT)

2. For each RMU, if there was a reception error, the message content is ineligible, or the RMU is not trusted, then the RMU is not an eligible voter.
   2.1. Eligible content for each RMU: RM(SPECIAL, INIT)
   2.2. Protocol checks:
      2.2.1. Expecting that at least one RMU is an eligible voter

3. Compute the Accept function for the messages received from the eligible voters.
   3.1. Cross-lane checks for each RMU:
      3.1.1. Expecting agreement with the result of the Accept function
   3.2. Protocol checks:
      3.2.1. Expecting agreement among a majority of the eligible voters

4. When the Accept output is asserted, broadcast RM(SPECIAL, ECHO), start the synchronization-reset timer, and send RM(SPECIAL, INIT) to the attached PE.

Process P3: RMUs

1. Receive the messages from the BIUs.
   1.1. Communication checks for each BIU:
      1.1.1. Expecting no link errors
   1.2. In-line checks for each BIU:
      1.2.1. Expecting reception within a predetermined local-time interval
      1.2.2. Expecting exactly one message
      1.2.3. Expected content: RM(SPECIAL, ECHO)

2. For each BIU, if there was a reception error, the message content is ineligible, or the BIU is not trusted, then the BIU is not an eligible voter.
   2.1. Eligible content for each BIU: RM(SPECIAL, ECHO)
   2.2. Protocol checks:
      2.2.1. Expecting that at least one BIU is an eligible voter

3. Compute the Accept function for the messages received from the eligible voters.
   3.1. Cross-lane checks for each BIU:
      3.1.1. Expecting agreement with the result of the Accept function
   3.2. Protocol checks:
      3.2.1. Expecting agreement among a majority of the eligible voters

4. When the Accept output is asserted, broadcast RM(SPECIAL, ECHO) and start the synchronization-reset timer.

Process P4: BIUs

1. Receive the messages from the RMUs.
  1.1. Communication checks for each RMU:
    1.1.1. Expecting no link errors
  1.2. In-line checks for each RMU:
    1.2.1. Expecting reception within a predetermined local-time interval
    1.2.2. Expecting exactly one message
    1.2.3. Expected content: RM(SPECIAL, ECHO)

2. For each RMU, if there was a reception error, the message content is ineligible, or the RMU is not trusted, then the RMU is not an eligible voter.
  2.1. Eligible content for each BIU: RM(SPECIAL, ECHO)
  2.2. Protocol checks:
    2.2.1. Expecting that at least one BIU is an eligible voter

3. Compute the Accept function for the messages received from the eligible voters.
  3.1. Cross-lane checks for each RMU:
    3.1.1. Expecting agreement with the result of the Accept function
  3.2. Protocol checks:
    3.2.1. Expecting agreement among a majority of the eligible voters

In processes P1 through P4, a reception error or a cross-lane check error is sufficient evidence to accuse the corresponding node of the opposite kind.

In processes P1 through P4, it is expected that at least one node of the opposite kind is eligible to vote and also to have agreement on received message timing for a majority of the eligible voters. An error detection by any of these checks is an indication of a clique failure.

5.4. Collective Diagnosis

In the Collective Diagnosis mode, the clique executes the Collective Diagnosis protocol to achieve a consistent diagnostic view of every ROBUS node in the system, including those that are not part of the clique. Two executions of the protocol are performed: one to diagnose RMUs and another to diagnose BIUs. Each protocol execution takes the accusations against each defendant of a particular kind from all of the nodes that are part of the clique, combines them to assess the trustworthiness of each defendant, and then distributes the resulting conviction results. Both executions of the Collective Diagnosis protocol use ROBUS messages formatted to carry diagnostic data corresponding to accusations or convictions. This message format is presented in Section 3. The Collective Diagnosis protocol is a synchronous protocol implemented using synchronous communication.

The Collective Diagnosis protocol was originally inspired the MAFT approach to on-line diagnosis

presented in [Walter 97]. Geser and Miner [Geser 04] developed the current version of the protocol, which is optimized for the ROBUS.

Note that the Collective Diagnosis protocol is executed by nodes in each of the major modes, except for the Self-Test mode.

5.4.1. Collective Diagnosis protocol for RMU defendants

Figure 5.6 shows the message flow graph for the Collective Diagnosis protocol applied to diagnose RMU defendants. The labels next to the arrows indicate the type of data transmitted by the message sources.

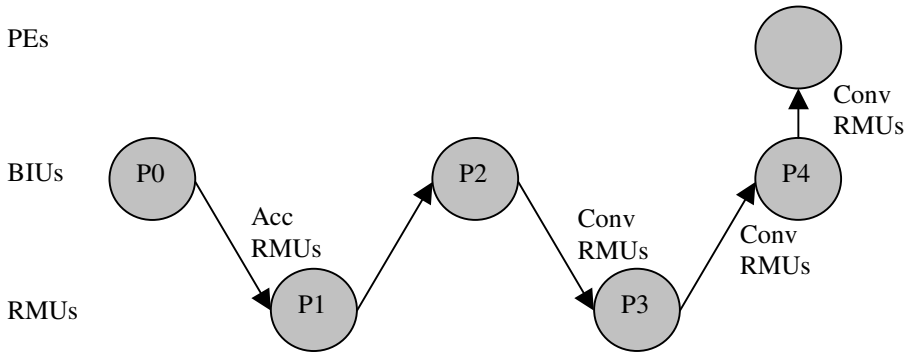


Figure 5.6: Message flow graph for the Collective Diagnosis protocol for RMU defendants

The description of the protocol is presented next. The merge operation in process P1 is a two-input Boolean OR function.

Process P0: BIUs
1. Broadcast the local accusations against the RMUs.

Process P1: RMUs
1. Receive the messages from the BIUs.
  1.1. Communication checks for each BIU:
    1.1.1. Expecting no link errors
  1.2. In-line checks for each BIU:
    1.2.1. Expecting reception within a predetermined local-time interval
    1.2.2. Expecting exactly one message
    1.2.3. Expected content: RM(DATA, *)

2. For each BIU, if there was a reception error, the message content is ineligible, or the BIU is not trusted, then the BIU is not an eligible voter.

2.1. Eligible content for each BIU: RM(DATA, *)
2.2. Protocol checks:
2.2.1. Expecting that at least one BIU is an eligible voter

3. For each RMU defendant, perform a bit vote on the accusations received from eligible voters.

4. For each RMU defendant, merge the result of the bit vote with the local accusation value.

5. Broadcast the results of the merge operations as a single ROBUS message.

Process P2: BIUs

1. Receive the messages from the RMUs.
1.1. Communication checks for each RMU:
1.1.1. Expecting no link errors
1.2. In-line checks for each RMU:
1.2.1. Expecting reception within a predetermined local-time interval
1.2.2. Expecting exactly one message
1.2.3. Expected content: RM(DATA, *)

2. For each RMU, if there was a reception error, the message content is ineligible, or the RMU is not trusted, then the RMU is not an eligible voter.
2.1. Eligible content for each RMU: RM(DATA, *)
2.2. Protocol checks:
2.2.1. Expecting that at least one RMU is an eligible voter

3. For each RMU defendant, perform a bit vote on the accusations received from eligible voters.

4. Broadcast the results of the bit vote operations as a single ROBUS message.

Process P3: RMUs

1. Receive the messages from the BIUs.
1.1. Communication checks for each BIU:
1.1.1. Expecting no link errors
1.2. In-line checks for each BIU:
1.2.1. Expecting reception within a predetermined local-time interval
1.2.2. Expecting exactly one message
1.2.3. Expected content: RM(DATA, *)

2. For each BIU, if there was a reception error, the message content is ineligible, or the BIU is not trusted, then the BIU is not an eligible voter.
2.1. Eligible content for each BIU: RM(DATA, *)
2.2. Protocol checks:
2.2.1. Expecting that at least one BIU is an eligible voter

3. Perform a word vote on the messages received from eligible voters.
  3.1. Cross-lane checks for each BIU:
    3.1.1. Expecting agreement with the result of the vote
  3.2. Protocol checks:
    3.2.1. Expecting agreement among a majority of the eligible voters

4. Broadcast the result of the vote.

5. The result of the vote has the updated convictions against the RMUs.
  5.1. Protocol checks:
    5.1.1. Expecting that not all of the RMUs are convicted
  5.2. Self-check:
    5.2.1. Is the local node convicted?

Process P4: BIUs

1. Receive the messages from the RMUs.
  1.1. Communication checks for each RMU:
    1.1.1. Expecting no link errors
  1.2. In-line checks for each RMU:
    1.2.1. Expecting reception within a predetermined local-time interval
    1.2.2. Expecting exactly one message
    1.2.3. Expected content: RM(DATA, *)

2. For each RMU, if there was a reception error, the message content is ineligible, or the RMU is not trusted, then the RMU is not an eligible voter.
  2.1. Eligible content for each RMU: RM(DATA, *)
  2.2. Protocol checks:
    2.2.1. Expecting that at least one RMU is an eligible voter

3. Perform a word vote on the messages received from eligible voters.
  3.1. Cross-lane checks for each RMU:
    3.1.1. Expecting agreement with the result of the vote
  3.2. Protocol checks:
    3.2.1. Expecting agreement among a majority of the eligible voters
    3.2.2. Is the the result of the vote equal to the result in process P2?

4. Send the result of the vote to the attached PE.

5. The result of the vote has the updated convictions against the RMUs.
  5.1. Protocol checks:
    5.1.1. Expecting that not all of the RMUs are convicted

In processes P1 through P4, a reception error is sufficient evidence to accuse the corresponding node of the opposite kind.

In processes P2 through P4, it is expected that at least one node of the opposite kind is eligible to vote. In processes P3 and P4, it is expected to have agreement on received message content for a majority of the eligible voters. An error detection by any of these checks is an indication of a clique failure.

The cross-lane checks in processes P3 and P4 compare the result of the vote with the received input from each node of the opposite kind. An error detection by these checks is sufficient evidence to accuse the corresponding node of the opposite kind.

In processes P3 and P4, it is expected that the result of the word vote does no indicate that all of the RMUs are convicted. A violation of this expectation indicates a clique failure.

The self-check in process P3 is stated as a question because the expected result depends on the mode of a node executing this protocol process. For a node in Clique Preservation mode or in the first pass in the Clique Join mode, the result of the vote should indicate a conviction against the node. For a node in Clique Initialization, Clique Preservation, or the second pass in the Clique Join mode, the expected result is that the node is not convicted. In Clique Detection mode, a detected error is interpreted as a clique failure. In all of the other cases, an error detection indicates a local failure or a clique failure.

Nodes in the Clique Join, Clique Initialization, and Clique Preservation modes expect that, for each defendant, the results in processes P2 and P4 are equal. An error detection in any of these modes indicates a clique failure. For nodes in the Clique Detection mode, there is no expected relation between the results.

5.4.2. Collective Diagnosis protocol for BIU defendants

Figure 5.7 shows the message flow graph for the Collective Diagnosis protocol applied to diagnose BIU defendants. The labels next to the arrows indicate the type of data transmitted by the message sources.

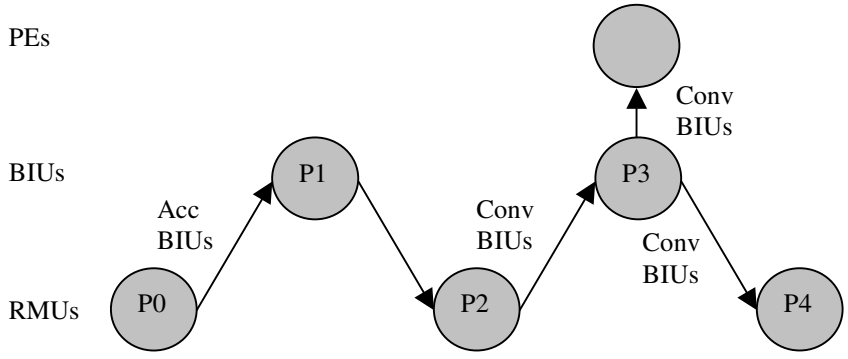


Figure 5.7: Message flow graph for the Collective Diagnosis protocol for BIU defendants

The description of the protocol is presented next.

Process P0: RMUs

1. Broadcast the local accusations against the BIUs as a single diagnostic message.

Process P1: BIUs

1. Receive the messages from the RMUs.
   1.1. Communication checks for each RMU:
      1.1.1. Expecting no link errors
   1.2. In-line checks for each RMU:
      1.2.1. Expecting reception within a predetermined local-time interval
      1.2.2. Expecting exactly one message
      1.2.3. Expected content: RM(DATA, *)

2. For each RMU, if there was a reception error, the message content is ineligible, or the RMU is not trusted, then the RMU is not an eligible voter.
   2.1. Eligible content for each RMU: RM(DATA, *)
   2.2. Protocol checks:
      2.2.1. Expecting that at least one RMU is an eligible voter

3. For each defendant BIU, perform a bit vote on the accusations received from eligible voters.

4. For each defendant BIU, merge the result of the bit vote with the local accusation.

5. Broadcast the results of the merge operations as a single diagnostic message.

Process P2: RMUs

1. Receive the messages from the BIUs.
   1.1. Communication checks for each BIU:
      1.1.1. Expecting no link errors
   1.2. In-line checks for each BIU:
      1.2.1. Expecting reception within a predetermined local-time interval
      1.2.2. Expecting exactly one message
      1.2.3. Expected content: RM(DATA, *)

2. For each BIU, if there was a reception error, the message content is ineligible, or the BIU is not trusted, then the BIU is not an eligible voter.
   2.1. Eligible content for each BIU: RM(DATA, *)
   2.2. Protocol checks:
      2.2.1. Expecting that at least one BIU is an eligible voter

3. For each BIU defendant, perform a bit vote on the accusations received from eligible voters.

4. Broadcast the results of the bit vote operations as a single diagnostic message.

Process P3: BIUs

1. Receive the messages from the RMUs.
   1.1. Communication checks for each RMU:
      1.1.1. Expecting no link errors
   1.2. In-line checks for each RMU:
      1.2.1. Expecting reception within a predetermined local-time interval
      1.2.2. Expecting exactly one message
      1.2.3. Expected content: RM(DATA, *)

2. For each RMU, if there was a reception error, the message content is ineligible, or the RMU is not trusted, then the RMU is not an eligible voter.
   2.1. Eligible content for each RMU: RM(DATA, *)
   2.2. Protocol checks:
      2.2.1. Expecting that at least one RMU is an eligible voter

3. Perform a word vote on the messages received from eligible voters.
   3.1. Cross-lane checks for each RMU:
      3.1.1. Expecting agreement with the result of the vote
   3.2. Protocol checks:
      3.2.1. Expecting agreement among a majority of the eligible voters

4. Broadcast the result of the vote.

5. Send the result of the vote to the attached PE.

6. The result of the vote has the updated convictions against the BIUs.
   6.1. Protocol checks:
      6.1.1. Expecting that not all of the BIUs are convicted
   6.2. Self-check:
      6.2.1. Is the local node convicted?

Process P4: RMUs

1. Receive the messages from the BIUs.
   1.1. Communication checks for each BIU:
      1.1.1. Expecting no link errors
   1.2. In-line checks for each BIU:
      1.2.1. Expecting reception within a predetermined local-time interval
      1.2.2. Expecting exactly one message
      1.2.3. Expected content: RM(DATA, *)

2. For each BIU, if there was a reception error, the message content is ineligible, or the BIU is not trusted, then the BIU is not an eligible voter.
   2.1. Eligible content for each BIU: RM(DATA, *)
   2.2. Protocol checks:
      2.2.1. Expecting that at least one BIU is an eligible voter

3. Perform a word vote on the messages received from eligible voters.
  3.1. Cross-lane checks for each BIU:
    3.1.1. Expecting agreement with the result of the vote
  3.2. Protocol checks:
    3.2.1. Expecting agreement among a majority of the eligible voters
    3.2.2. Is the the result of the vote equal to the result in process P2?

4. The result of the vote has the updated convictions against the BIUs.
  4.1. Protocol checks:
    4.1.1. Expecting that not all of the BIUs are convicted

This protocol is essentially the same as the protocol for collective diagnosis of RMUs. All the comments stated in the previous section for the error checks of that protocol apply to this one as well.

5.4.3. Concurrent diagnosis for RMU and BIU defendants

It is possible to diagnose the RMU and the BIU defendants concurrently by overlapping the message flow graphs for the executions of the Collective Diagnosis protocol for RMU and BIU defendants. Figure 5.8 shows the resulting pattern. This pattern achieves a significant reduction in the time required to complete the self-diagnosis of the bus.

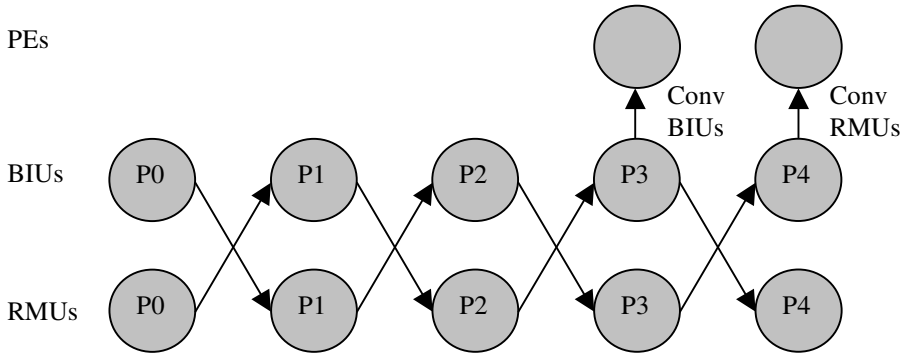


Figure 5.8: Message flow graph for the concurrent diagnosis of RMU and BIU defendants

6. Self-Test

The Self-Test mode serves two purposes. First, it establishes a checkpoint in which the nodes are required to exercise and assess the status of their circuitry before attempting to join other nodes on the bus. The effectiveness of this activity in stopping faulty nodes is dependent on the fault coverage provided by the self-test procedures, which are implementation-dependent. In a non-developmental version of the ROBUS, the results of the self-test would be included in the decision of a node to transition to a disabled state permanently, or to remain active and try to return to normal operation. For the current version of the ROBUS, a failure of the self-test is considered a local failure that should trigger a re-entry into the Self-Test mode. If the self-test can detect a particular permanent-fault condition affecting a node, the node will remain in the Self-Test mode indefinitely.

The second purpose of the Self-Test mode is to provide a safe state to which the ROBUS nodes can go after detecting a failure and before attempting to re-engage. A detected failure can be caused by phenomena external to the bus (e.g., lightning or HIRF) that can have an unknown duration and can affect multiple nodes simultaneously. The nodes do not have the means to accurately determine the cause, duration, or number of nodes affected by a fault-causing phenomenon. Because of this, worst-case conditions are always assumed. The nodes are programmed to disengage from the bus as soon as a failure is detected and then run a self-test continuously for a time interval at least as long as the worst-case duration of a fault-causing phenomenon plus the worst-case failure detection delay. This behavior ensures that the fault-causing phenomenon has subsided and the affected nodes have disengaged from the bus by the time a node exits the Self-Test mode.

Figure 6.1 shows the operations performed in the Self-Test mode. A ROBUS node enters this mode during a startup after the power-on enable or during a restart after the detection of a local node failure or a bus failure. The first action is to reset completely and disable the output. It is assumed that a fault can propagate to all the components within an FCR. Therefore, when a fault is detected, the state data of the node is considered corrupted, and none of it is carried over, irrespective of whether it is a local failure or a bus failure. The disabling of the output ensures that any remaining members of the clique and other nodes trying to rejoin the bus will consistently detect that the node is untrustworthy. The next action in the Self-Test mode is to execute the self-test until the bus has settled. This duration is called the **Upset Abatement Delay**. Appendix G examines the recovery process in detail and shows how to compute this delay.

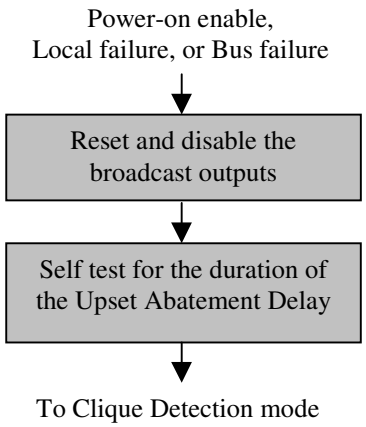


Figure 6.1: Activities during the Self-Test mode

7. Clique Detection

After exiting the Self-Test mode, a node needs to determine if there is a clique operating in the Clique Preservation mode or if a new clique needs to be formed. In this version of the ROBUS, the approach selected is to assume that there is clique present on the bus and try to acquire its state while simultaneously monitoring for any indications that the clique is not valid. Figure 7.1 shows the minor-mode transition graph. A recovering node is unsynchronized when it enters the Clique Detection mode. In the Local Diagnosis Acquisition mode, the recovering node observes the nodes of the opposite kind in order to determine a trusted set operating in the Clique Preservation mode. Because it is unsynchronized, the recovering node can only use its local time for coarse assessment of the timing characteristics of other nodes. In the Synchronization Acquisition mode, the recovering node synchronizes to the clique leveraging the accumulated diagnostic observations. Local-time synchronization enables the recovering node to perform refined timing observations and to gather state data from the synchronous protocols. In the Collective Diagnosis Acquisition mode, the recovering node gets the conviction results computed by the clique. The information collected up to this point is considered sufficient for the recovering node to make a final determination about the presence of a clique. If it has evidence that a clique is present, the recovering node will proceed to the Clique Join mode to attempt to become a member of the clique. Otherwise, it will transition to the Clique Initialization mode to form a new clique.

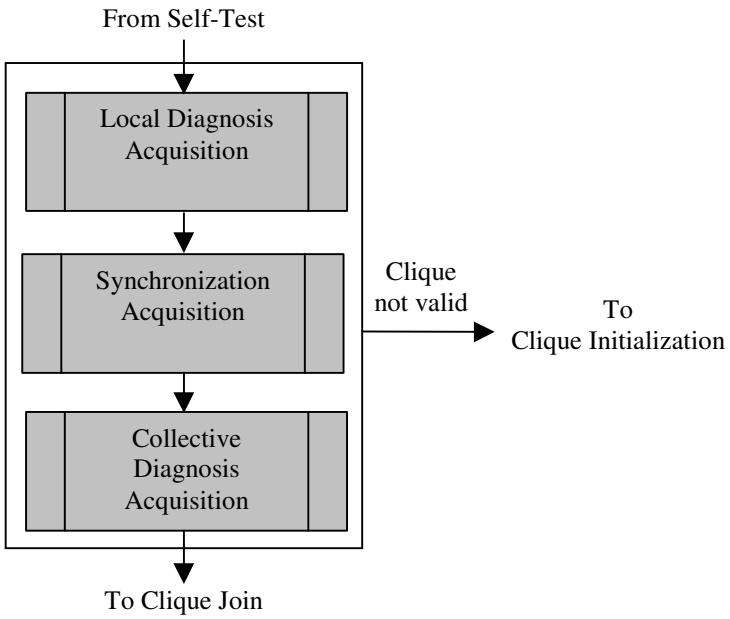


Figure 7.1: Minor-mode transitions for Clique Detection mode

The only protocol-independent condition that indicates a local failure of the recovering node is the assertion of an accusation against itself. Each of the following protocol-independent conditions indicates a clique failure: number of trusted BIUs equal to zero, and number of trusted RMUs equal to zero. Protocol-dependent conditions corresponding to a clique failure are described with the corresponding protocols.

7.1. Local Diagnosis Acquisition

During Local Diagnosis Acquisition, a recovering node determines a trusted set of opposite kind nodes operating in the Clique Preservation mode. For this version of the ROBUS, this is accomplished by separate in-line monitors that check the received message patterns for each opposite kind node. Many combinations of timing and content checks are possible. There is a tradeoff between the effectiveness of the checks and their complexity. In general, a high degree of effectiveness requires a high degree of design refinement, which results in a more complex implementation. The chosen approach is considered a reasonable balance between the effectiveness and the complexity of the checks.

The local diagnosis is performed in two phases. In the first phase, each in-line monitor searches for an ECHO message from its corresponding source node. Since the sources are assumed to be in Clique Preservation, they are expected to broadcast an ECHO message once per re-synchronization period. If a particular monitor does not receive an ECHO within the expected time interval, the corresponding source is accused. The monitors that receive the ECHO message transition to the second phase of diagnosis in which the content sequence for received messages is checked. An ECHO message is the last one in the Synchronization Preservation protocol. After that, an inline monitor expects to receive the messages for the Collective Diagnosis, Schedule Update, and PE Communication protocols. For Collective Diagnosis, the expected number of messages is constant and only DATA messages are expected. For Schedule Update, the expected number of messages is also constant and only PE ERROR or DATA messages are allowed. For PE Communication, the number of messages can vary from zero up to a known maximum and there are only a few valid message formats. The reception of an INIT message signals the arrival at the Synchronization Preservation protocol. The INIT should be followed by an ECHO message. If the received message pattern was not valid or the second ECHO is not received within the time of one re-synchronization period, the source node can be accused.

7.2. Synchronization Acquisition

After identifying a preliminary set of trusted nodes, the next step is to synchronize to the clique. Figure 7.2 shows the required activities. During Frame Synchronization, a recovering node achieves coarse synchronization by distinguishing received synchronization messages from different executions of the Synchronization Preservation protocol. During Synchronization Capture, the ECHO messages of the Synchronization Preservation protocol are used to tightly synchronize to the clique.

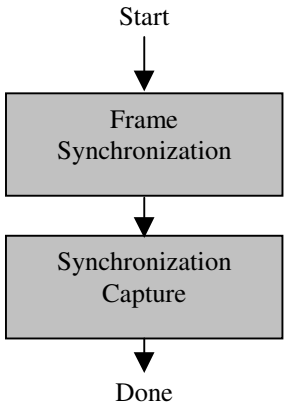


Figure 7.2: Activities for Synchronization Acquisition

7.2.1. Frame Synchronization

The purpose of Frame Synchronization is to ensure that the Accept function in Synchronization Capture receives the ECHO messages from trustworthy sources participating in the Synchronization Preservation protocol. Achieving Frame Synchronization is equivalent to finding the time gap between consecutive executions of the Synchronization Preservation protocol. The Frame Synchronization protocol for this version of the ROBUS is presented below. The protocol monitors the ECHO messages from the trusted nodes identified during Local Diagnosis Acquisition. The time interval measured by the gap timer corresponds to the maximum observed relative skew between received ECHO messages from trustworthy nodes. The calculation of this skew is described in Appendix C. The protocol has provisions for handling untrustworthy trusted nodes. In addition, the in-line checks can be used to identify untrustworthy nodes. Any opposite kind node that violates the expectations can be accused.

Process:
1. Start the gap timer.

2. While the gap timer has not expired:
  2.1. If an error is detected, remove the corresponding source from the eligible sources.
    2.1.1. Communication checks for each opposite-kind source:
      2.1.1.1. Expecting no link errors
    2.1.2. In-line checks for each opposite-kind source:
      2.1.2.1. At most one RM(SPECIAL, ECHO) message expected before the gap timer expires.

  2.2. If an RM(SPECIAL, ECHO) message is received from an eligible source, then restart the gap timer.

3. Done

7.2.2. Synchronization Capture

During Synchronization Capture, a recovering node synchronizes to the clique by applying the Accept function to ECHO messages received from trusted sources. Figure 7.3 shows the message flow graph for the Synchronization Preservation protocol augmented to include the P3C and P4C Synchronization Capture processes. Recovering RMUs execute process P3C, and recovering BIUs execute process P4C. A recovering BIU sends an ECHO message to its PE. The use of an ECHO message rather than an INIT message allows the PE to easily recognize that these are different synchronization events with different associated timing.

The P3C and P4C processes are described next. The processes are triggered by the end of the Frame Synchronization protocol. Because the recovering nodes are only loosely synchronized, they do not have precise expectations about the time of arrival of the ECHO messages. Any messages received that are not ECHO messages must be ignored by the protocol. No reception checks are performed to determine the eligible voters. These are equal to the trusted nodes of the opposite kind at the start of the protocol. The only significant difference in the descriptions for P3C and P4C is that the BIUs in process P4C must send an ECHO message to their PEs. Agreement checks are performed by comparing the time between

relevant events against an expected duration. Appendix C provides additional information about the timing aspects of this protocol, including the calculation of the delays measured by the synchronization-reset timers in processes P3C and P4C.

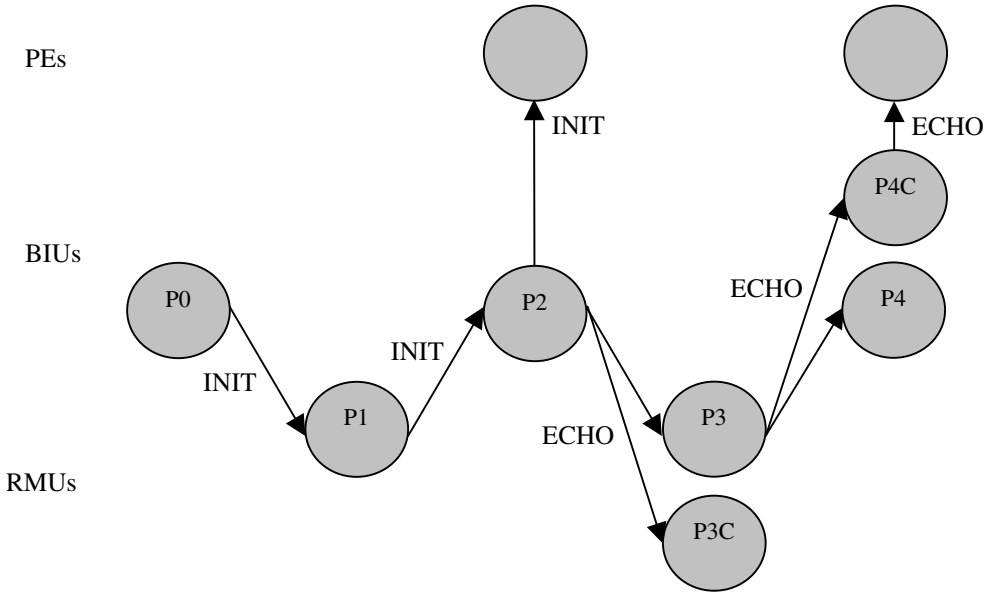


Figure 7.3: Message flow graph for the Synchronization Preservation protocol with Synchronization Capture processes

Process P3C: RMUs
1. Receive the RM(SPECIAL, ECHO) messages from the BIUs.

2. Compute the Accept function for the messages received from the eligible voters.
  2.1. Cross-lane checks for each BIU:
    2.1.1. Expecting agreement with the result of the Accept function
  2.2. Protocol checks:
    2.2.1. Expecting agreement among a majority of the eligible voters

3. When the Accept output is asserted, start the synchronization-reset timer.

Process P4C: BIUs
1. Receive the RM(SPECIAL, ECHO) messages from the RMUs.

2. Compute the Accept function for the messages received from the eligible voters.
  2.1. Cross-lane checks for each RMU:
    2.1.1. Expecting agreement with the result of the Accept function
  2.2. Protocol checks:
    2.2.1. Expecting agreement among a majority of the eligible voters

3. When the Accept output is asserted, start the synchronization-reset timer and send RM(SPECIAL, ECHO) to the attached PE.

A cross-lane check error in processes P3C or P4C is sufficient evidence to accuse the corresponding node of the opposite kind.

In processes P3C and P4C, it is expected to have agreement on received message timing for a majority of the eligible voters. An error detection by any of these checks is an indication of a clique failure.

7.3. Collective Diagnosis Acquisition

During Collective Diagnosis Acquisition mode, a recovering node gets the conviction results computed by the clique. To do this, the recovering node executes the Collective Diagnosis protocols as if it were part of the clique, with only one exception. For recovering BIUs executing the Collective Diagnosis protocol for RMU defendants, there is no expectation that the result in process P4 should equal the result in process P2. Similarly, for recovering RMUs executing the Collective Diagnosis protocol for BIU defendants, there is no expectation that the result in process P4 should equal the result in process P2.

8. Clique Join

During the Clique Join mode, a recovering node demonstrates that it is suitable for admission to the clique. For this version of the ROBUS, that consists of operating properly for one full diagnostic cycle.

Figure 8.1 shows the flow of operation for the Clique Join mode. When the recovering node enters this mode, its time and diagnostic state variables are in agreement with the corresponding state variables of the clique. This enables the recovering node to internally operate as if it were a member of the clique. The only significant difference is that its output is disabled. The node should enable its output just before the clique starts gathering local diagnostic observations for the next local diagnostic cycle, which occurs at the beginning of Collective Diagnosis. Once the output has been enabled, the recovering node should expect to be admitted to the clique after the following diagnostic cycle. This is confirmed by results of the Collective Diagnosis protocol indicating that the node is not convicted. If so, the recovering node transitions to the Clique Preservation mode to operate as a member of the clique. However, if the node is convicted, this is sufficient evidence to conclude that the node or the clique has failed. Therefore, the recovering node should transition to the Self-Test mode.

Each of the following protocol-independent conditions indicates a local failure or a bus failure: number of trusted BIUs equal to zero, number of trusted RMUs equal to zero, and assertion of an accusation against self. The protocol-dependent conditions are described with the corresponding protocols.

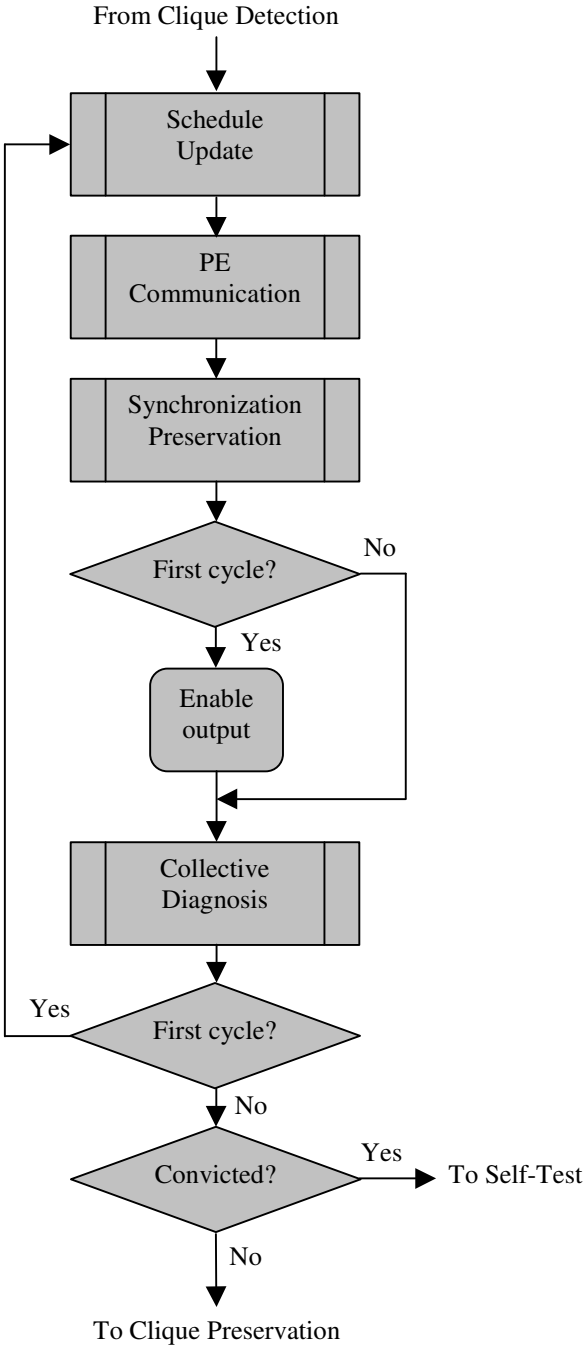


Figure 8.1: Minor-mode transitions for the Clique Join mode

9. Clique Initialization

A node transitions from the Clique Detection mode to the Clique Initialization mode after determining that there is not a valid clique operating in the Clique Preservation mode. The purpose of this major mode is to form a new clique. A node operates in this mode with the presumption that there is a group of nodes trying to form a clique and that all of them enter this mode in the unsynchronized state with a known bound on the relative local-time skew. To form a new clique, a set of nodes trying to form the clique must be identified, and local time and diagnostic state agreement must be achieved. This is done while simultaneously monitoring for indications that a clique cannot be formed.

Figure 9.1 shows the main activities performed in the Clique Initialization mode. None of the time and diagnostic state gathered in the Clique Detection mode is valid once a node transitions to this mode. The first step is to clear all the state data and enable the output to allow communication with other nodes. In the Initial Diagnosis mode, a preliminary set of nodes trying to form a clique is found. In Initial Synchronization, the uncertainty in the local-time synchronization is reduced to the level expected in the synchronized state. In Collective Diagnosis, the nodes reach agreement on the membership of the clique.

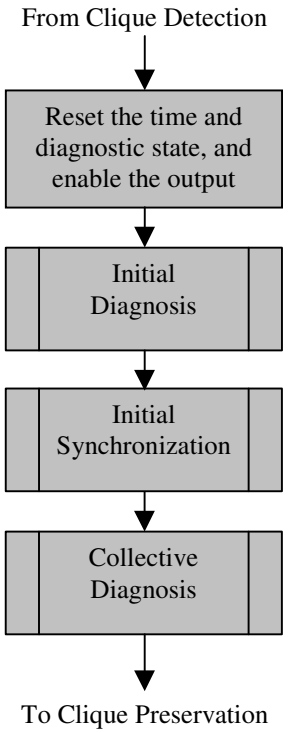


Figure 9.1: Minor-mode transitions for the Clique Initialization mode

Each of the following protocol-independent conditions indicates a local failure or a bus failure for the Clique Initialization mode: number of trusted BIUs equal to zero, number of trusted RMUs equal to zero, and assertion of an accusation against self. The protocol-dependent conditions are described with the corresponding protocols.

9.1. Initial Diagnosis

In the Initial Diagnosis mode, the nodes execute a synchronous protocol to determine an initial trusted set taking advantage of the known bound on the synchronization precision when operating in the unsynchronized state. Figure 9.8 shows the message flow graph. Notice that there is no visibility to the behavior of nodes of the same kind. Thus, a particular node can only assess nodes of the opposite kind.

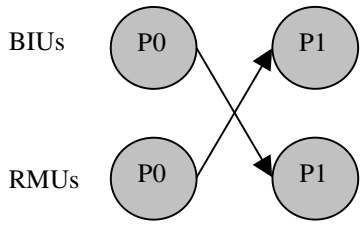


Figure 9.8: Message flow graph for the Initial Diagnosis protocol

The description of the protocol follows. A reception error in process P1 is sufficient evidence to accuse the corresponding node of the opposite kind.

Process P0: BIUs and RMUs
1. Broadcast RM(SPECIAL, CLIQUE_INITIALIZATION).

Process P1: BIUs and RMUs
1. Receive the messages from the nodes of the opposite kind.
  1.1. Communication checks for each node of the opposite kind:
    1.1.1. Expecting no link errors
  1.2. In-line checks for each node of the opposite kind:
    1.2.1. Expecting reception within a predetermined local-time interval
    1.2.2. Expecting exactly one message
    1.2.3. Expected content: RM(SPECIAL, INITIALIZATION)

9.2. Initial Synchronization

The purpose of the local-time synchronization protocol executed in this mode is to reduce the relative skew to the level required for normal synchronized operation. The protocol is triggered a fixed delay after the completion of the Initial Diagnosis protocol. The Initial Synchronization protocol is based on the same basic protocol as the Synchronization Preservation protocol, and it can handle any specified bound on the initial relative local-time skew, even one much larger than the final skew.

Figure 9.2 shows the message flow graph for the protocol. The protocol is an agreement generation protocol using the fixed-delay model for point-to-point communication. The labels near the arrows indicate the type of SPECIAL message that is transmitted by the sending process. Notice that in process P4 the BIUs send to the PEs an ECHO message rather than an INIT message, which is used only by the Synchronization Preservation protocol.

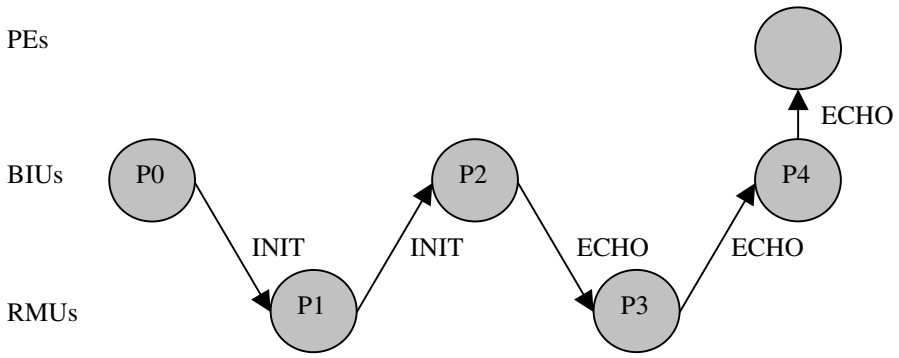


Figure 9.2: Message flow graph for the Initial Synchronization protocol

The description of the protocol is presented next. Because the bound on the initial relative local-time skew can be large, the nodes do not have precise expectations about the time of arrival of the messages. The processes ignore any received messages that are not of the expected kind, and no reception error checks are performed in any of the processes. The eligible voters are equal to the trusted nodes of the opposite kind at the start of the protocol. The communication and process delays for Initial Synchronization must be approximately equal to the ones for the Synchronization Preservation protocol in order for the final synchronization precision to be the same. This can result in a situation in which the protocol delay from P0 to the synchronization reset in P3 and P4 is much smaller than the initial time skew. Since the Accept functions assert their outputs shortly after receiving more than half of the expected inputs, it is possible that the ECHO messages are broadcast before some trustworthy nodes are ready to send their INIT messages in processes P0 and P1. In addition, since the objective of the protocol is to generate reference ECHO events to trigger the enabling of the synchronization-reset timers, there is no need for a node to send an INIT message after it has sent an ECHO message. The blocking of INITs after ECHOs ensures that, in effect, the INIT-sending processes terminate after the completion of the ECHO-sending processes. Appendix C provides additional information about the timing aspects of this protocol.

Process P0: BIUs

1. If it is time to send and RM(SPECIAL, ECHO) has not been broadcast, then broadcast RM(SPECIAL, INIT).

Process P1: RMUs

1. Receive the RM(SPECIAL, INIT) messages from the BIUs.

2. Compute the Accept function for the messages received from the eligible voters.

3. When the Accept output is asserted, if RM(SPECIAL, ECHO) has not been broadcast, then broadcast RM(SPECIAL, INIT).

Process P2: BIUs
1. Receive the RM(SPECIAL, INIT) messages from the RMUs.

2. Compute the Accept function for the messages received from the eligible voters.

3. When the Accept output is asserted, broadcast RM(SPECIAL, ECHO).

Process P3: RMUs
1. Receive the RM(SPECIAL, ECHO) messages from the BIUs.

2. Compute the Accept function for the messages received from the eligible voters.
  2.1. Cross-lane checks for each BIU:
    2.1.1. Expecting agreement with the result of the Accept function
  2.2. Protocol checks:
    2.2.1. Expecting agreement among a majority of the eligible voters

3. When the Accept output is asserted, start the synchronization-reset timer and broadcast RM(SPECIAL, ECHO).

Process P4: BIUs
1. Receive the RM(SPECIAL, ECHO) messages from the RMUs.

2. Compute the Accept function for the messages received from the eligible voters.
  2.1. Cross-lane checks for each RMU:
    2.1.1. Expecting agreement with the result of the Accept function
  2.2. Protocol checks:
    2.2.1. Expecting agreement among a majority of the eligible voters

3. When the Accept output is asserted, start the synchronization-reset timer and send RM(SPECIAL, ECHO) to the attached PE.

A cross-lane check error in processes P3 or P4 is sufficient evidence to accuse the corresponding node of the opposite kind.

In processes P3 and P4, it is expected to have agreement on received message timing for a majority of the eligible voters. An error detection by any of these checks is an indication of a clique failure.

9.3. Collective Diagnosis

The Collective Diagnosis protocol is described in Section 5.

10. Concluding remarks

ROBUS is the central feature of the SPIDER IMA architecture currently in the research and technology development phase. The purpose of this work is to design a flexible architecture that can be configured to satisfy a wide range of performance and reliability requirements. It is envisioned that ultimately the ROBUS will be a family of communication systems based on a common theory of fault-tolerant distributed computation and communication. ROBUS-2 is a concept-demonstration version of the ROBUS intended for laboratory experimentation and demonstrations of the capabilities. This section summarizes the attributes of ROBUS-2 presented in the previous sections. This is followed by an overview of some of the current ideas that may be explored to develop the concept.

10.1. ROBUS-2

ROBUS-2 provides four basic services to the PEs: message broadcast, communication schedule update, time reference, and self-diagnosis. The message broadcast service is realized by the PE Broadcast protocol, which is a Byzantine Agreement protocol that ensures result consistency even if the source PE or the source BIU is arbitrarily faulty. Communication schedule update uses the Schedule Update protocol to provide PE-fault tolerant and ensure schedule-update consistency at the RMUs, BIUs, and PEs. The time reference service uses the Synchronization Preservation protocol, an agreement protocol that generates precise periodic reference events used by RMUs, BIUs, and PEs to synchronize their local-time clocks. Self-diagnosis is realized by the internal ROBUS diagnostic system, which consists of local and collective diagnostic processing to assess the status of individual nodes and the bus as a whole. The Collective Diagnosis protocol is an agreement protocol that processes the local diagnostic assessments and establishes a consistent view of the status of every ROBUS node. The appendices present the supporting fault-tolerance theory for these services, including the fault conditions that guarantee the expected results.

Time-triggered operation is enabled by the periodic execution of the synchronization preservation protocol. The known precision bound on the relative local-time skew is exploited by using distributed synchronous composition to coordinate distributed operations. The synchronous communication model solves the basic problem of transferring data between independently clocked nodes while maintaining overall timing predictability in the execution of distributed protocols.

Enforcement of the scheduled bus access pattern for the PE-message broadcast service is relatively simple given the system topology and the use of time-triggered operation. There are three layers of protection against unauthorized bus access. The first line of enforcement is at the source BIUs, which are expected to forward the messages of their attached PEs at the scheduled time only. The second layer is realized by the routing function at each RMU, which is programmed to relay the messages from a particular PE at the scheduled time only. The final enforcement layer is at the receiving BIUs, which use exact-majority, dynamic word voting to filter out messages relayed by untrustworthy RMUs.

The diagnostic system is designed based on a transient-fault model for the ROBUS nodes. All node faults are considered transient and, accordingly, a node is never permanently removed from the system. A single instance of detected misbehavior is considered sufficient evidence to remove a node from the trusted set. Readmission into the trusted set is allowed once a full error-free diagnostic cycle has been completed. This is a conservative diagnostic policy.

Dynamic voting is the main mechanism for neutralizing undiagnosed node failures. Node trust and

voter eligibility are reassessed for each voting protocol operation using the latest collective diagnosis results as well as the local diagnostic information in order to achieve a maximum reconfiguration rate that is much faster than the rate of execution of the Collective Diagnosis protocol. Input-error detection, dynamic voting, fail-stop node behavior, and fast reconfiguration allow a clique to continue service delivery to the PEs for some scenarios in which a large number of nodes become untrustworthy within a short time interval.

For ROBUS-2, the only difference between startup and restart is the response-triggering event. The node recovery procedure is independent of whether the trigger is a power-on enable, a local failure, or a bus failure. The Self-Test mode provides some assurance that a node is operating properly before it attempts to recover. In the Clique Detection mode, a recovering node attempts to acquire the state of a clique while monitoring for indications that one is not present. Successful acquisition of the state is followed by a transition to the Clique Join mode to demonstrate that it is suitable for admission into the clique. A recovering node in the Clique Detection mode transitions to the Clique Initialization mode if it detects that there is not a valid clique operating in the Clique Preservation mode. Node diagnosis in order to establish which nodes can be trusted is the most basic function for fault-tolerant operation in the ROBUS distributed system. Local Diagnosis Acquisition and Initial Diagnosis compute initial trusted sets for the Clique Detection and Clique Initialization modes, respectively. In addition to having a trusted set, the nodes must synchronize their local-time clocks before being able to communicate efficiently. The Synchronization Acquisition mode protocols provide the means for a recovering node to synchronize to an existing clique. The Initial Synchronization protocol allows recovering nodes to synchronize to form a new clique. With synchronized operation established, the nodes are able to use more rigorous error detection and diagnosis, thus strengthening the fault-tolerance of a clique. Appendix G examines the startup and restart capabilities.

The ability of a ROBUS clique to maintain coordination independently of PE failures is realized by the PE-error checks, which report detected PE errors to the BIUs, and more importantly, by the agreement generation properties of the Schedule Update and PE Broadcast protocols. Appendices D and E examine the properties of these protocols.

10.2. ROBUS-X

ROBUS-1 was a proof-of-concept design meant to demonstrate some basic features of the ROBUS. That version, presented in [Miner 02], had a simple initialization mechanism, no failure recovery capability, relatively simple error detection and diagnosis, and a static communication schedule (i.e., loaded off-line). The scheme for collective diagnosis relied on an interactive-consistency message-broadcast protocol to ensure diagnosis agreement, but that resulted in a heavy performance penalty. Furthermore, the implementation allowed at most one PE message to be “in transit” through the bus at any one time. This sequential end-to-end processing limitation severely restricted the maximum message throughput of the bus.

ROBUS-2 is another proof-of-concept design meant to demonstrate a combination of attributes, including robust fault tolerance, high message throughput, and a dynamically updateable communication schedule (i.e., loaded on-line). ROBUS-2 incorporates much of the theoretical insight into the operation of the ROBUS fault-tolerance protocols gained since the first version was designed. A hardware realization of the BIU and RMU nodes currently under development achieves a maximum message throughput of one PE message per clock tick by pipelining the processes of the PE Broadcast protocol. Ultimately, the throughput of the bus measured in messages per clock tick will be limited by the

throughput of the communication links. That realization of ROBUS-2 will be implemented on a COTS-based platform with the ROBUS functions programmed on field-programmable gate-arrays (FPGAs). Fault injection in a number of environments, including VHDL simulations, high-intensity radiated fields (HIRF) and neutron particle radiation, will likely be used to assess the robustness of the fault-tolerance and to gain additional insight for further development.

The following ideas may be explored for further development of ROBUS.

- For ROBUS-2, the PE bus access sequence is determined by the fixed identification numbers. This constraint can be easily relieved by augmenting the schedule update to include an access-sequence number for each PE. The PEs would access the bus in a Round Robin with respect to the dynamically assigned access-sequence numbers, rather than their fixed identification numbers. This is a much more flexible access pattern.

- ROBUS-2 uses a simple message format: (Tag, Payload), with a payload of one word. A more flexible and efficient communication system may be achieved with a variable-length message format with one or more words per message.

- A ROBUS-2 clique operating in the Clique Preservation mode delivers services in a fixed cyclic sequence. This sequence was chosen because it is a simple representative pattern that fulfills the basic requirement of making these services available to the PEs, and because it simplifies the state acquisition sequence in the Clique Detection mode. With this pattern, all the services operate at the same rate. For real applications, additional consideration should be given to the service timing requirements of the PEs, including the PE-to-PE communication requirements. A study should be conducted to determine the best service-delivery sequence taking into consideration the requirements of the ROBUS and the PEs.

- The diagnostic system of ROBUS-2 only recognizes nodes as relevant diagnosable elements. The links are considered to be parts of the nodes. The result is that a node can be accused or convicted when messages are not propagating properly though a majority of its input or output links, even if the node itself is computing properly and many of its links are in good condition. One way to improve the fault tolerance of the ROBUS may be to increase the granularity of the diagnostic system to consider nodes and links, and then modifying the protocols to exploit the additional diagnostic information. The preferred redundancy management strategy for some applications is to use whatever resources are available in order to continue service delivery (i.e., “never give up”).

- The design of the diagnostic system of ROBUS-2 is based on two principles: any evidence of misbehavior is sufficient to distrust a node, and one diagnostic cycle without errors is sufficient to re-establish trust. The main constraint for the inclusion of error checks and diagnostic rules in the design was to ensure compliance with the required properties of correctness and agreement on non-asymmetric defendants. This approach is adequate for a technology development activity meant to assess and demonstrate the potential capabilities of ROBUS, but the result may be a design that is excessively complex for real-world applications. Furthermore, such a simple diagnostic policy may not yield optimum reliability or the highest probability of tolerating correlated transient-fault events. For future versions of the ROBUS, we intend to explore the use of diagnostic policies based on the on-line diagnosis algorithms presented in [Walter 97], especially algorithm HD, which can be tuned to differentiate between permanent and transient faults. Studies on the relation between diagnostic policies and the resulting attributes for ROBUS (e.g., reliability, survivability, cost, etc.) would be useful in guiding further development activities. The SPIDER reliability study presented in

[Latronico 04] is a good starting point. In addition to addressing high-level attributes, studies should address issues like the selection of error checks and the rules for mapping detected errors to node trust, voter eligibility, and other diagnostic system outputs.

- The time kept by ROBUS-2 is not synchronized to external events. In essence, the Synchronization Preservation and Initial Synchronization protocols take as inputs a distributed event generated at the BIUs and compute new distributed events with higher precision bounds. If synchronization to an external time reference is required by the PEs, they must realize that capability independently from the ROBUS time service.

  The ROBUS synchronization protocols can be modified as follows to synchronize the bus and the PEs to an external event: the external event is read by the PEs, which generate messages that are sent to the BIUs using fixed-delay communication; once the BIUs receive these messages, they generate INIT messages just like for the current protocols; the remaining operations of the protocols remain as they are. Having the bus synchronized to an important external event enables timely interaction between the PEs and the external world.

- The number of point-to-point communication links for the ROBUS topology increases with the number BIUs and RMUs. For a system with N BIUs and M RMUs, the total number of one-directional links for BIU-RMU communication is 2NM. Including the PE-BIU links, the total number of one-directional links grows to 2N(M + 1). If the links are implemented using bidirectional communication cables, the required number of cables can be reduced to NM for the BIU-RMU communication links, or N(M + 1) including PE-BIU links. The wiring for BIU-RMU communications can be reduced by using one-to-many broadcast links driven by a single transmitter at each BIU and each RMU. In that case, the number of cables can be reduced to N+M for the BIU-RMU communication links, or 2N + M overall. A study should be conducted to determine the advantages and disadvantages of various wiring options.

- The total amount of cabling required (in terms of total aggregate length) can be reduced by using the configuration depicted in Figure 10.1. In this configuration, BIUs and RMUs are in close proximity to one another forming what is essentially a fault-tolerant hub. The PEs may be placed in widely distributed locations. The total number of individual communication paths remains unchanged, but the amount of cabling can be significantly less than for a configuration in which the BUIs and RMUs are farther away from one another.

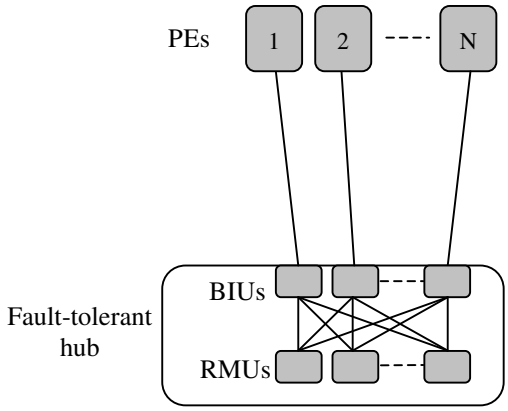


Figure 10.1: ROBUS in a fault-tolerant hub configuration

- The configuration shown in Figure 10.1 emphasizes of the PE-BIU links. One idea to develop the ROBUS is to explore the design of PE-BIU interfaces based on common commercial communication links (e.g. IEEE-1394 Firewire). Additional logic would have to be added at the PE and BIU ends to complement such links in order to realize all the functionality required for operation in the ROBUS (e.g., message format translation, clock synchronization, error detection, and BIU interfacing).

- ROBUS-2 was designed to operate with at most one active clique on the bus at any given time. No assurance of any kind is given about the behavior of the system if this condition is violated. It is known that if there are two or more mutually exclusive cliques simultaneously active on the bus, and the nodes in one clique distrust the nodes in the other cliques, then the system can remain in that state indefinitely. For some applications it is important to have assurance that such a multi-clique condition is impossible or extremely improbable. Under relatively benign conditions such that the BUIs and RMUs fail mainly because of physical degradation or random localized transient faults, an existing clique should be able to remain active and attract recovering nodes. The probability that a new clique does not form, or that multiple ones do, is likely to be higher when the bus is exposed to harsh conditions that can overwhelm the fault tolerance of a clique. For some applications, avoidance of harsh conditions is a practical way to ensure that the system remains safely within its fault tolerance limits. For other applications, the ROBUS must be capable of reliably re-establishing a single clique. If the assumptions of the Clique Initialization mode (see Section 9 and Appendices C and G) are guaranteed to be satisfied, the restart approach in ROBUS-2 is adequate. Otherwise, some other means to enable the ROBUS to return to normal operation, preferably on its own, will be necessary. At this time we do not have protocols that allow the ROBUS to return to normal operation from an arbitrary state (i.e., self-stabilize).

- ROBUS-2 nodes of a particular kind (i.e., BIU or RMU) are differentiated only by their assigned identification numbers. In every other respect, the nodes are considered to be part of a single uniform group. One way to improve the design is to divide the nodes into a core group and a client group. The core group would be composed of all the RMUs and a subset of the BIUs. The core nodes and their corresponding PEs would be responsible for computing the basic bus functions like synchronization, collective diagnosis, and communication schedule update. The computed results for these functions would be broadcast to the client BIUs, whose main tasks would be to provide access to the bus to additional PEs. The client BIUs would also have to gather diagnostic information and send it to the core group for the collective diagnosis. This allocation of functions would reduce the complexity of the client BIUs, and possibly the RMUs as well. [Kopetz 87] proposed a similar approach to implement a clock synchronization service in generic distributed real-time systems.

- The ROBUS enables the development of several fault-tolerance strategies combining simplex PE nodes. Figure 10.2 presents a sample configuration with three PEs in a triple modular redundant (TMR) configuration (labeled vPE 0, or virtual Processing Element 0), four processors in a dual-dual configuration (vPE 2), and a single simplex processor (vPE 1). For ROBUS-2, the bus interacts with the PEs as if they were part of a single uniform group, and the distributed SPIDER operating system manages the PE configuration. Note that ROBUS-2 supports having only a subset of the PEs compute the communication schedule. Since PE_ERROR messages are not eligible inputs to the Schedule Update protocol, the PEs that are not going to send a schedule update to the bus can remove themselves from consideration by signaling an error to their BIUs, which would then send PE_ERROR messages. In addition, note that the design of ROBUS-2 is compatible with dynamic reconfiguration strategies at the PE level, in which the processors get reassigned to particular virtual PEs in real time as operating modes change or failures occur. The bus is completely unaware of such rearrangements.

Future concept-development efforts could explore moving some PE-configuration support functions to the bus. For example, consider a system in which PE-communication scheduling is done with respect to the virtual PEs. For the configuration in Figure 10.2 during the PE Communication mode, the ROBUS could vote on the fly messages simultaneously sent by the PEs in vPE 0 and then broadcast the results. For vPE 2, the ROBUS could monitor messages simultaneously sent by dual PEs 2 and 7 and broadcast the messages from one of them so long as a discrepancy is not detected between the two input streams. If a discrepancy were detected during the transmission, the ROBUS would immediately switch to dual PEs 4 and 5. Note that the updating of the communication schedule could also take advantage of these features by using one predetermined virtual PE to compute the schedule and then having the ROBUS vote only on the inputs from the corresponding PEs. A system with such features would have a faster response to PE faults and reduced communication bandwidth requirement. In order to support these configuration-dependent functions, the ROBUS would have to have information about how the PEs are configured. A protocol similar to the Schedule Update protocol can be used to download such information from the PEs to the ROBUS.

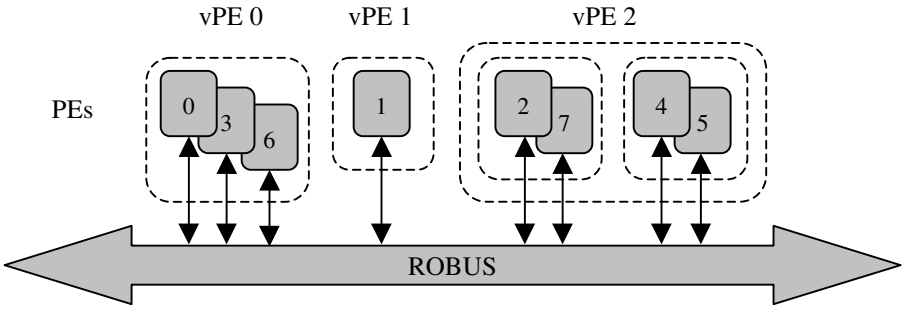


Figure 10.2: Sample PE configuration

- Figure 10.3 illustrates another concept for configuring the PEs. Here the PEs and their BIUs are divided into groups based on common attributes (e.g., implementing a particular function). Each group is composed of one or more PEs. The RMUs support independent broadcast communication within each group, as well as inter-group communication. Thus, the RMUs implement multicast communication with one or more simultaneous input message streams and each stream relayed to one or more groups. This configuration can be visualized as being the result of merging multiple basic ROBUS systems with corresponding RMUs linked by a routing function. Such a combined system is more capable than a single ROBUS-2 system, but it is also more complex. The time reference service can be easily implemented using one group as a core and the rest as clients that are synchronized with respect to the core. The scheduling of messages would have to take into consideration inter-group messages. Diagnosis, both local and collective, would be modified to use policies that exploit knowledge of how the BIUs and PEs are grouped.

Other areas that may be explored for further development include ways to explicitly support event-triggered messages with strict end-to-end latency constraints, and efficient ways of implementing the ROBUS system in hardware and/or software. Future versions of ROBUS, generically designated as ROBUS-X, will likely have more refined designs geared toward practical applications with strict cost and complexity constraints.

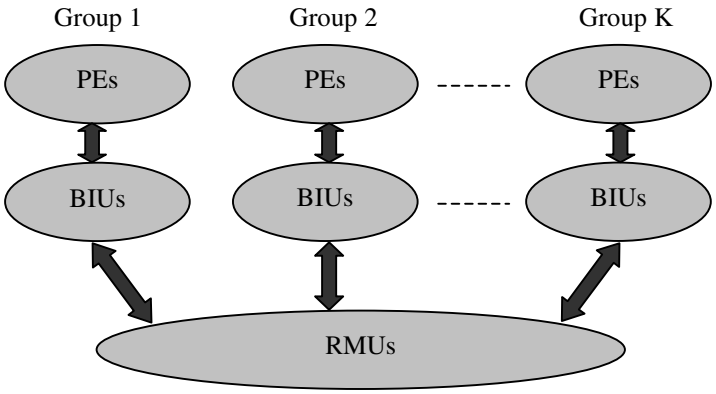


Figure 10.3: PE groups with multicast ROBUS communication

Appendix A. ROBUS fault-tolerance fundamentals

This appendix describes the basic theory of distributed computation applied to the design of the ROBUS protocols for the delivery of fault-tolerant services to the PEs.

The presentation in the following three subsections uses concepts adapted from the material in [Avizienis 04], [Laprie 95], and [Suri 95].

A.1. Faults, errors, and failures

In general, a system has a specification and is composed of multiple components or sub-systems. Likewise, each of these sub-systems has a specification and an internal structure of interconnected components. In what follows, it is assumed that the specifications at every hierarchical level are correct and free from ambiguity, omission, and other kinds of defects.

The terms fault, error, and failure are used to describe a cause-and-effect relationship between undesired circumstances in the context of the hierarchical composition of a system. A **failure** occurs when the behavior of a system fails to provide the desired service. Failure is assessed at the external interface of a system and is determined by deviations from the behavior expected according to the specification. An **error** is a deviation from the intended value and/or timing of data (including signals and state variables) at a particular hierarchical level. A **fault** is a defect in a system component.

The fault, error, and failure terms facilitate the structured analysis of the failure characteristics of a system and the determination of failure causality chains from low-level components to higher-level components. In a simple chain, the failure of a system is due to the presence of an error in the system. This error is caused by a faulty component that failed to deliver the expected service. At this point, the component can be seen as a system and the failure causality chain expanded by further exploring the hierarchical structure. The chain ends when a component is reached beyond which no internal structure can be discerned or is of interest.

A.2. Fault characteristics

The following fault classification criteria were considered during the design process of the ROBUS.

A.2.1. Cause

The causes of faults can be divided into two main categories: design faults and physical faults. **Design faults** are specification or implementation mistakes such that the system does not function as desired. The fault-tolerance capabilities of ROBUS are not meant to handle design faults. Instead, formal analysis and other design verification activities are used to minimize their introduction (for example, see [Geser 04] and [Pike 04]). **Physical faults** are caused by internal defects and external disturbances. Examples of internal defects include manufacturing imperfections, component wear-out, internal electromagnetic interference (EMI) (e.g., cross-talk and ground bounce), and radioactive impurities in semiconductor parts. External disturbances that can cause faults include particle radiation, external EMI (e.g., lightning, high-intensity radiated fields, and electrostatic discharge), input power fluctuations, and environmental extremes (e.g., temperature, vibration, and shock). Part of the design process of a system is to assess and

control the time rate of occurrence of physical faults. Examples of ways to manage the physical fault rates include component selection, shielding, environmental qualification testing, reliability testing, control of the operational environment, and preventive maintenance.

A.2.2. Correlation and extent

Physical faults on two separate components are **independent** if there is no causal or common cause relation between them. Otherwise, the faults are **correlated**. In real systems, faults can propagate from one component to another and they can be caused by the same instance of a particular phenomenon, especially when exposed to external disturbances that can reach multiple components. Thus, complete independence between faults occurring at separate components is not entirely possible. The **extent** of a fault refers to the number of affected components. For the ROBUS topology, this can range from one node to every node on the bus. Note that the PEs can be affected by the same phenomena as the BIUs and RMUs. The ROBUS is designed to handle scenarios involving a large number of nodes simultaneously becoming faulty.

A.2.3. Activity

A component is said to have experienced a fault when its behavior deviates from its specification. A fault is said to be **latent** or **dormant** if it has not affected the externally observed behavior of the component. Once an error occurs at the interface of the component, the fault is said to be **active**. Similarly, an error is latent when it has not propagated to the system interface, and it becomes active once it causes a failure. This concept is used when considering the effects of faults at the interface of individual BIU, RMU, and PE nodes, as well as at the interface of the ROBUS viewed as a unit.

A.2.4. Duration

Permanent faults appear and remain present in a system until they are removed through a maintenance action. **Transient faults** appear for a limited duration of time. Depending on the structure and behavior of a system, both permanent and transient faults can cause errors and failures of permanent or transient duration. The fault type of main interest for this developmental version is the single-event transient that is active for a bounded duration of time. Nevertheless, the redundancy management system of the ROBUS can handle many forms of permanent and transient faults.

A.2.5. Consistency of perception

Consistency of perception refers to the degree to which observations of the fault manifestations differ among the direct observers. A fault is **symmetric** if all properly working observers receive agreeing inputs. Otherwise, the fault is called **asymmetric**. Consistency of perception is an important fault characteristic for the ROBUS since asymmetric manifestations can threaten the integrity of a clique by causing divergence in distributed computation results. The ROBUS protocols are designed to handle a bounded number of simultaneously active asymmetric faulty nodes.

A.2.6. In-line detectability

If a direct observer can detect input errors caused by a fault using independent in-line observations (i.e., without comparison against data from redundant sources), then the observer can take appropriate actions to prevent the propagation of the errors to the computation results. Input error detection by means of communication and in-line checks enables the ROBUS nodes to identify and remove erroneous data from consideration in local protocol operations. In general, an increase in the percentage of in-line detectable faults results in a reduction of the ROBUS failure probability.

A.2.7. Diagnosability

In the context of the ROBUS, a fault is diagnosable if the properly working observers of the affected node can identify the node as a source of errors. The diagnosability of a fault is limited by the error detection and diagnosis capabilities of the observers, as well as by the message flow patterns and types of operations required by the distributed protocols. The ROBUS nodes handle diagnosable faults by removing offending nodes from the trusted set. In general, an increase in the percentage of diagnosable faults results in a reduction of the ROBUS failure probability.

A.3. Fault and error containment

The most basic element of the fault-tolerance strategy of the ROBUS is the fault containment region (FCR). The purpose of the FCR is to ensure a high degree of independence between physical faults occurring in different system components. Each BIU and RMU must be in a separate FCR. The building of FCRs requires careful consideration of the physical characteristics of a system to ensure that a proper degree of containment is present for all coupling paths. Examples of ways to achieve fault independence include the use of independent power supplies for each FCR, independent cooling systems, separate electromagnetic shielding for each FCR, fiber-optic data links, and physical distancing of the FCRs. In practical terms, it is impossible to achieve complete fault independence for all possible faults. A special concern is faults caused by external EMI, like lightning and high intensity electromagnetic fields. This kind of phenomenon has the potential to engulf a system causing simultaneous faults in multiple FCRs. Shielding and other techniques can be used to minimize the threat to a system, but total elimination of the threat is not always possible. The goal in real systems is to achieve a degree of fault independence between FCRs that is acceptable for the application.

Error containment is the second layer of the fault-tolerance strategy of the ROBUS. The error containment mechanisms of the ROBUS are aimed at preventing the propagation of errors between FCRs. The ultimate goal of the redundancy management strategy of the ROBUS is to prevent the propagation of errors to the external interfaces of the bus at the BIUs. Given that each PE is attached to a single BIU, which could be faulty, the bus is considered to have experienced a failure only when errors reach the interface of a fault-free BIU that is a clique member. The ROBUS performs an internal failure assessment by monitoring its own activity searching for violations to the conditions that ensure the effective performance of the error containment mechanisms. Such a violation is an indication that error containment cannot be guaranteed.

A.4. Node health and inclusion status

Due to the diagnostic and recovery capabilities of the ROBUS, the analysis must take into consideration more than just the fault status of the nodes. A node can be operating according to its specification, yet, if it is recovering, it may not be ready to deliver the services expected by a clique. In addition, even if a node can be relied upon, the clique is not able to integrate it immediately because trust is only asserted at the boundaries of collective diagnostic intervals. On the other hand, a clique may not have enough evidence to remove a particular untrustworthy node whose behavior is not covered by the error detection and diagnosis capabilities of the system.

The following criteria determine the health and inclusion status of a node.

- **Goodness**: A node is **good** if it behaves according to its specification. Otherwise, the node is **bad** or **faulty**.

- **Trustworthiness**: A node is **trustworthy** if it is suitable to participate in the delivery of services to the PEs. Otherwise, the node is **untrustworthy**. A trustworthy node must be good and its state must be in agreement with the state of other good clique members.

- **Diagnostic status**: A node is **trusted** if it is a clique member, and thus allowed to participate in the execution of distributed operations. Otherwise, the node is **distrusted**.

A.5. Fault model

For the analysis of the ROBUS, a distributed system in which a group of nodes collaborate to achieve common goals, it is more useful to know how the behavior of a node is perceived at the receivers (or direct observers) than what actually occurs at the output of the node itself. In this section, first, the behavior of a node is classified according to its manifestations at the direct observers for a given transmission, and then this classification is leveraged to define a general node fault model. This model is a modified version of the hybrid fault-effect model presented by Thambidurai and Park in [Thambidurai 88].

A.5.1. Instantaneous behavioral manifestations

For this model, the classification of the behavior of a node applies to one message transmission, expected or unexpected, from the node to its trustworthy direct observers. As used here, the validity of the behavior of a node depends on the specific activity (e.g., process P2 of the Collective Diagnosis protocol in the Clique Preservation mode) being carried out by the trustworthy direct observers, which are either clique members or, in the case of recovering observers, are in state agreement with a clique. A transmission (or a non-transmission, if one is not expected, according to the specification) from a given node is valid if it is functionally equivalent to the behavior expected from a trustworthy node of the same kind. Both the timing and the content characteristics of a transmission are important determinants of the perceived behavior at the direct observers.

The following categories are mutually exclusive and, taken together, cover all possible behavioral manifestations.

- **Valid**: The behavior of the node perceived by the trustworthy direct observers is in accordance with the specification.

- **Manifest**: The node does not behave as expected by its trustworthy direct observers and this is detected by all of them using their input error detectors.

- **Symmetric**: The node does not behave as expected by its trustworthy direct observers, which receive consistent invalid inputs from the node, but fail to detect the misbehavior of the node using their input error detectors.

- **Asymmetric**: The node does not behave as expected by its trustworthy direct observers, which receive inconsistent inputs from the node, and some or all of them fail to detect the misbehavior of the node using their input error detectors.

The classification of a given physical fault is dependent on the particular input error detectors used by the trustworthy direct observers at the time of the message. For example, if there are no input error detectors active on the direct observers, then a fault may be classified as symmetric or asymmetric, but not manifest. Likewise, a fault that is classified as symmetric or asymmetric for one set of active error detectors may be classified as manifest for a different set of detectors.

In addition to the error detectors, the membership of the set of trustworthy direct observers is a critical determinant of the classification of a fault. For example, a fault is classified as asymmetric if only one trustworthy direct observer receives an undetected inconsistent input, but it would be classified as either manifest or symmetric if that particular observer became untrustworthy and everything else remained the same.

There is not a one-to-one relation between symmetry at the transmitter (i.e., consistent generation of a message) and consistency at the observers (i.e., consistent reception of a message). For example, it is also possible that the trustworthy direct observers receive the same message but disagree on the result of input error detection. This can happen if there is a timing violation in the transmission of the message such that not all of the observers determine that the received message arrived within the expected time interval. Note that an omissive node failure in which a node does not transmit an expected message always has symmetric manifestations. This is one of the reasons for designing the ROBUS nodes to disable their outputs upon detection of a failure, rather than sending some sort of "I_AM_FAULTY" message that could have asymmetric manifestations.

A.5.2. Node fault model

The node fault model separates the nodes into two main categories: trustworthy and untrustworthy. A node is untrustworthy if it is faulty or its state disagrees with the state of the trustworthy nodes. The externally perceived behavior of an untrustworthy node is dependent on factors like the characteristics of the fault affecting the node, the internal design of the node, the specific activity being carried out by the clique, and the specific input error detectors used by its trustworthy direct observers. The fault model avoids these complications by defining behavioral categories based on sets of instantaneous behavioral manifestations and allowing the behavior of the nodes to vary within the range of the corresponding set. This approach simplifies the abstract analysis of the bus, from which guidelines are then derived for the design of the nodes. The categories for the node fault model are the following.

- **Trustworthy**: The node is good and it can be counted upon to correctly deliver the expected services. The node exhibits only valid instantaneous behavioral manifestations.

- **Benign**: The node is untrustworthy with valid or manifest instantaneous behavioral manifestations.

- **Symmetric**: The node is untrustworthy with valid, manifest, or symmetric instantaneous behavioral manifestations.

- **Asymmetric**: The node is untrustworthy with valid, manifest, symmetric, or asymmetric instantaneous behavioral manifestations.

In addition, the timing of transitions between trustworthy and untrustworthy are modeled as follows.

- A node can transition from trustworthy to untrustworthy at any time.

- A node can transition from untrustworthy to trustworthy only at boundaries of the collective diagnostic intervals.

A.6. Basic design of the ROBUS protocols

The ROBUS protocols perform diverse functions. However, most of the protocols are based on the unified fault-tolerance protocol presented in [Miner 04]. This section describes the basic concepts used to design the protocols.

The ROBUS protocols are composed of one or two processing phases, each one having of two computation stages. A **stage** refers to the transmission of a message from one or more nodes of a particular kind to nodes of the opposite kind, and it involves the Send Process of the source nodes, their broadcast transmission links, and the Receive and Computation Processes at the nodes of the opposite kind. A **phase** is a complete message flow cycle in which messages are sent by a particular set of nodes, processed by nodes of the opposite kind, and then the results are returned to the first nodes for additional processing. Figure A.1 illustrates these concepts. The nodes are labeled simply as LEFT and RIGHT since, due to the symmetries of the ROBUS topology and of the basic protocol design concepts, it is not significant in the basic theory to know which are BIUs and which are RMUs. (Stages 2 and 3 form a third phase not shown in Figure A.1.)

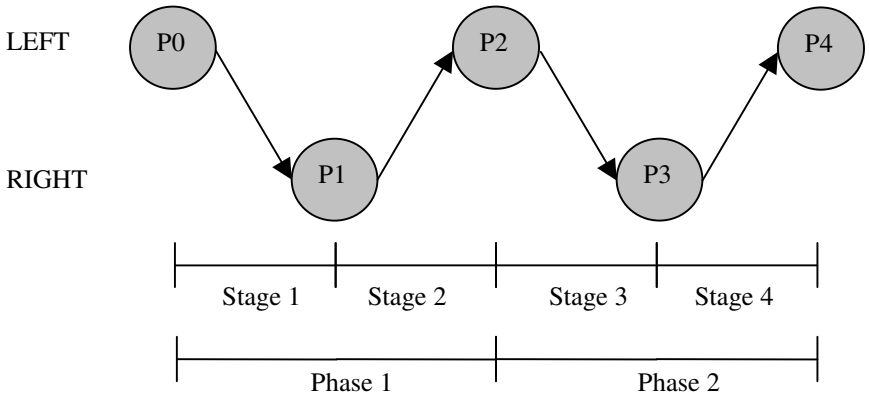


Figure A.1: Illustration of protocol stages and phases

A.6.1. Properties of protocol stages

Individual stage operations are the building blocks of the ROBUS protocols. In general, a stage consists of the transmission of data from one or more **source nodes** to **receiving nodes** of the opposite kind, followed by the application of a voting function to reduce the received data to a single value, which is the result of the stage operation. The ROBUS nodes use dynamic voting, in which only a selected group of inputs is considered in the voting operation. The sources whose inputs are allowed to participate in the vote are called the **eligible voters**. The eligibility conditions for a particular stage operation depend on the purpose of the protocol, the operation performed by the stage, and the position of the stage in the protocol.

The fundamental theory of operation of the ROBUS protocols is based on the middle-value-select voting function. Let E denote the number of eligible voters. For this version of the ROBUS, the middle value is given by the input in the $\lceil (E + 1)/2 \rceil$-th position when the eligible input values are arranged in order from minimum to maximum.

Two models of communication can be used to transmit data: exact and inexact. Let v_s denote the value transmitted by a trustworthy source, and let v_r denote the value received by a trustworthy receiver. In the **exact communication** model, each of the trustworthy receiving nodes receives the same value transmitted by a trustworthy source (i.e., $v_r = v_s$). In the **inexact communication** model, the transmission introduces uncertainty in the values received by the trustworthy nodes. The trustworthy receiving nodes do not necessarily get the same value transmitted by a trustworthy source, but the received value at each receiver satisfies the following constraint: $v_s - \varepsilon_l \leq v_r \leq v_s + \varepsilon_h$, where ε_l and ε_h denote the low-side and high-side error bounds (i.e., the error bounds in the negative and positive directions), respectively. The bound on the total communication imprecision, denoted by e, is $\varepsilon_l + \varepsilon_h$.

The behavior of untrustworthy sources must be taken into consideration to determine the results of stage operations. Only eligible voters can influence the voting results. A transmission from an untrustworthy eligible voter can have symmetric or asymmetric manifestations at the trustworthy receiving nodes of the opposite kind. The meaning of symmetric and asymmetric manifestations depend on the model of communication. For exact communication, a symmetric manifestation means that all of the trustworthy receivers get exactly the same arbitrary value. For inexact communication, a symmetric manifestation means that the trustworthy receivers get arbitrary values that differ from one another by at most e. An asymmetric manifestation for exact and inexact communication simply means that there are no relational constraints for the values received by trustworthy receivers. For some voter eligibility conditions, it is possible that the receivers do not agree on the eligibility of asymmetric sources.

The following subsections present some important properties of stage operations for the exact and inexact communication models.

A.6.1.1. Voting with exact communication

Let $v_{s,min}$ and $v_{s,max}$ denote the bounds for the values transmitted by the trustworthy sources, and let Δ denote the bound on the range of the values transmitted by the trustworthy sources: $\Delta = v_{s,max} - v_{s,min}$. $v_{r,i}$ denotes the voting result at receiving node i.

EV_i denotes the set of eligible voters at receiving node i. This set may be composed of trustworthy and untrustworthy sources. Twy_EV_i, Sym_EV_i, and $Asym\_EV_i$ denote the sets of trustworthy, symmetric untrustworthy, and asymmetric untrustworthy eligible voters at node i, respectively. It is

assumed that all of the trustworthy sources are eligible to vote at each trustworthy receiving node. In addition, it is assumed that the trustworthy receiving nodes agree on the eligibility of trustworthy and symmetric untrustworthy sources, but they may not agree on the eligibility of asymmetric sources. For the properties presented next, it is also assumed that the set of eligible voters at each trustworthy receiving node contains more trustworthy sources than untrustworthy ones. That is:

$|Twy\_EV_i| > |Sym\_EV_i| + |Asym\_EV_i|$,

where receiving node i is trustworthy, and |*| denotes the set cardinality function.

Validity: At each trustworthy receiver, the result of the voting function is in the interval $[v_{s,min}, v_{s,max}]$.

Proof: Conceptually, the middle-value-select voter at receiver i selects the value in the $\lceil (|EV_i| + 1)/2 \rceil$-th position from a list of eligible input values arranged in order from minimum to maximum. The assumption that at each trustworthy receiver the set of eligible voters contains more trustworthy sources than untrustworthy ones implies that at least $\lceil (|EV_i| + 1)/2 \rceil$ eligible input values (i.e., a majority) are in the interval $[v_{s,min}, v_{s,max}]$. Thus, at each trustworthy receiver there are fewer than $\lceil (|EV_i| + 1)/2 \rceil$ untrustworthy eligible voters. Although the values from untrustworthy eligible voters can be arbitrary, even if all of their values were less than $v_{s,min}$, there are not enough of them to cause the selection of a value smaller than $v_{s,min}$. Likewise, even if all of their values were larger than $v_{s,max}$, the selected value would be at most $v_{s,max}$. In general, the selection is a value from a trustworthy source, or the value from an untrustworthy source in the interval $[v_{s,min}, v_{s,max}]$.

Agreement propagation: The results of the middle-value-select voting functions at the trustworthy receivers differ from one another by at most Δ.

Proof: The validity property shows that the values selected at the trustworthy receivers are in the interval $[v_{s,min}, v_{s,max}]$, which corresponds to a total range of Δ. The actual agreement range can be smaller than this depending on the values received from untrustworthy eligible voters.

Agreement generation: If the sets of eligible voters at the trustworthy receiving nodes do not include asymmetric untrustworthy sources (i.e., $|Asym\_EV_i| = 0$ for each trustworthy receiving node i), then the voting results at the trustworthy receivers exactly agree.

Proof: If the sets of eligible voters at the trustworthy receivers do not include asymmetric sources, then the property of agreement for non-asymmetric defendants ensures that all the sets are identical. Therefore, the trustworthy receivers vote on the same set of values. Thus, the voting results will be the same.

A.6.1.2. Voting with inexact communication

For voting with inexact communication, the same assumptions are made as for voting with exact communication. The only significant difference here is the imprecision in the received values. The following properties are satisfied.

Validity: At each trustworthy receiver, the result of the voting function is in the interval $[v_{s,min} - \varepsilon_l, v_{s,max} + \varepsilon_h]$.

Proof: Since the minimum value transmitted by a trustworthy source is $v_{s,min}$, the minimum value

received by the trustworthy receivers from a trustworthy source is $v_{s,min} - \varepsilon_l$. Likewise, the maximum value transmitted is $v_{s,max}$, and the maximum value received from a trustworthy source is $v_{s,max} + \varepsilon_h$. Similarly to the case of voting with exact communication, the middle-value-select voter at receiver i selects the value in the $\lceil (|EV_i| + 1)/2 \rceil$-th position from a list of eligible input values arranged in order from minimum to maximum. In addition, at least $\lceil (|EV_i| + 1)/2 \rceil$ eligible input values (i.e., a majority) are in the interval $[v_{s,min} - \varepsilon_l, v_{s,max} + \varepsilon_h]$, and there are fewer than $\lceil (|EV_i| + 1)/2 \rceil$ untrustworthy eligible voters. Although the values from untrustworthy eligible voters can be arbitrary, even if all of their values were less than $v_{s,min} - \varepsilon_l$, there are not enough of them to cause the selection of a value smaller than $v_{s,min} - \varepsilon_l$. Likewise, even if all of their values were larger than $v_{s,max} + \varepsilon_h$, the selected value would be at most $v_{s,max} + \varepsilon_h$. In general, the selection is a value from a trustworthy source, or the value from an untrustworthy source in the interval $[v_{s,min} - \varepsilon_l, v_{s,max} + \varepsilon_h]$.

Agreement propagation: The results of the middle-value-select voting functions at the trustworthy receivers differ from one another by at most $\Delta + e$.

Proof: The validity property shows that the values selected at the trustworthy receivers are in the interval $[v_{s,min} - \varepsilon_l, v_{s,max} + \varepsilon_h]$, which corresponds to a total range of: $(v_{s,max} + \varepsilon_h) - (v_{s,min} - \varepsilon_l) = \Delta + e$. The actual agreement range can be smaller than this depending on the values received from untrustworthy eligible voters.

Agreement generation: If the sets of eligible voters at the trustworthy receiving nodes do not include asymmetric untrustworthy sources (i.e., $|Asym\_EV_i| = 0$ for each trustworthy receiving node i), then the voting results at the trustworthy receivers agree within e.

Proof: If the sets of eligible voters at the trustworthy receivers do not include asymmetric sources, then all the sets are identical. In addition, the inexact communication model ensures that, for each eligible voter, any two trustworthy receivers receive values that differ by at most e. Let EV denote the set of eligible voters. The voters at the trustworthy receivers select the value in the $\lceil (|EV| + 1)/2 \rceil$-th position from a list of eligible input values arranged in order from minimum to maximum. Let node x be the trustworthy receiver that has the result with the smallest value, denoted by v_x. Thus, node x received values smaller than or equal to v_x from at least $\lceil (|EV| + 1)/2 \rceil$ eligible sources. The corresponding receptions at the other trustworthy receivers can have a maximum value of at most $v_x + e$. Therefore, the voting results at those trustworthy receivers will be smaller than or equal to $v_x + e$, but not smaller than v_x. Thus, the voting results at the trustworthy receivers will agree within e.

A.6.2. Properties of protocol phases

A protocol phase is composed of two consecutive protocol stages in which the results of the first stage determine the inputs of the second stage. For each phase in Figure A.1, the LEFT nodes are the initial data sources and the final receivers, and the RIGHT nodes are the intermediate receivers and sources. Phases 1 and 2 are called the **agreement generation phase** and the **agreement propagation phase**, respectively. The processes for the agreement generation phase are labeled P0, P1, and P2. For the agreement propagation phase, the processes are P2, P3, and P4. In what follows, we refer to processes P1, P2, P3, and P4 in Figure A.1 as **receiving processes**, in order to differentiate them from process P0, which does not involve the reception of messages.

Consider the agreement generation phase. The LEFT nodes execute processes P0 and P2, and the RIGHT nodes execute process P1. Let $EV_{P1,i}$ and $EV_{P2,j}$ denote the set of eligible voters in process P1 at

RIGHT node i and in process P2 at LEFT node j, respectively. It is assumed that the set of eligible voters at each trustworthy receiver of a particular kind includes all the trustworthy sources of the opposite kind. In addition, it is assumed that the trustworthy receiving nodes of a particular kind agree on the eligibility of trustworthy and symmetric untrustworthy sources of the opposite kind, but they may not agree on the eligibility of asymmetric sources. For the properties presented next, it is also assumed that the set of eligible voters at each trustworthy receiving node contains more trustworthy sources than untrustworthy ones. For process P1 at the RIGHT nodes:

$|Twy\_EV_{P1,i}| > |Sym\_EV_{P1,i}| + |Asym\_EV_{P1,i}|$,

for trustworthy receiving RIGHT node i. For process P2 at the LEFT nodes:

$|Twy\_EV_{P2,j}| > |Sym\_EV_{P2,j}| + |Asym\_EV_{P2,j}|$,

for trustworthy receiving LEFT node j. Similar assumptions are made for processes P3 and P4 of the agreement propagation phase.

In addition, it is assumed that at every trustworthy LEFT receiver or at every trustworthy RIGHT receiver there are no asymmetric eligible voters for any of the corresponding receiving processes. That is:

$|Asym\_EV_{P1,i}| = 0$ and $|Asym\_EV_{P3,i}| = 0$ for each trustworthy RIGHT receiver i, or

$|Asym\_EV_{P2,j}| = 0$ and $|Asym\_EV_{P4,j}| = 0$ for each trustworthy LEFT receiver j.

The following subsections present some important properties of phase operations for the exact and inexact communication models.

A.6.2.1. Agreement generation phase

Let $v_{P0,min}$ and $v_{P0,max}$ denote the bounds for the values transmitted by the trustworthy LEFT nodes for process P0. The value transmitted by process P1 is equal to the result of its vote. Thus, $v_{P1,min}$ and $v_{P1,max}$ denote the bounds for the voting results and the transmitted values for process P1 at the trustworthy RIGHT nodes. $v_{P2,min}$ and $v_{P2,max}$ denote the bounds for the voting result for process P2 at the trustworthy LEFT nodes. Δ_{P0}, Δ_{P1}, and Δ_{P2} denote the bounds on the range of the voting results at the trustworthy nodes for processes P0, P1, and P2, respectively.

Operations with the exact communication model are considered first, followed by operations with the inexact communication model.

A.6.2.1.1. Voting with exact communication

The following properties hold for an agreement generation phase with exact communication.

Validity: At each trustworthy LEFT node, the result of the vote in process P2 is in the interval $[v_{P0,min}, v_{P0,max}]$.

Proof: Based on the validity property for a stage operation with exact communication, the voting results for process P1 at trustworthy RIGHT nodes are in the interval $[v_{P0,min}, v_{P0,max}]$. The validity

property applied to the second stage ensures that the voting results for process P2 at trustworthy LEFT nodes are also in the interval $[v_{P0,min}, v_{P0,max}]$.

Agreement generation: The voting results for process P2 at the trustworthy LEFT nodes exactly agree (i.e., $\Delta_{P2} = 0$).

Proof: Two cases must be considered.

Case 1: $|Asym\_EV_{P1,i}| = 0$: According to the agreement generation property for a stage operation with exact communication, the voting results for process P1 at trustworthy RIGHT nodes will exactly agree (i.e., $\Delta_{P1} = 0$). The agreement propagation property ensures that the range is preserved by the second stage. Therefore, the voting results exactly agree.

Case 2: $|Asym\_EV_{P2,j}| = 0$: Based on the agreement propagation property for a stage operation, the voting results for process P1 at trustworthy RIGHT nodes agree within Δ_{P0} (i.e., $\Delta_{P1} = \Delta_{P0}$). The agreement generation property for a stage operation ensures that the voting results for process P2 at trustworthy LEFT nodes exactly agree.

A.6.2.1.2. Voting with inexact communication

The following properties hold for an agreement generation phase with inexact communication.

Validity: At each trustworthy LEFT node, the result of the vote in process P2 is in the interval $[v_{P0,min} - 2\varepsilon_l, v_{P0,max} + 2\varepsilon_h]$.

Proof: Based on the validity property for a stage operation with inexact communication, the voting results for process P1 at trustworthy RIGHT nodes are in the interval $[v_{P0,min} - \varepsilon_l, v_{P0,max} + \varepsilon_h]$. The validity property applied to the second stage ensures that the voting results for process P2 at trustworthy LEFT nodes are in the interval $[v_{P0,min} - 2\varepsilon_l, v_{P0,max} + 2\varepsilon_h]$.

Agreement generation: The voting results for process P2 at the trustworthy LEFT nodes agree within 2e (i.e., $\Delta_{P2} = 2e$).

Proof: Two cases must be considered.

Case 1: $|Asym\_EV_{P1,i}| = 0$: According to the agreement generation property for a stage operation with inexact communication, the voting results for process P1 at trustworthy RIGHT nodes will agree within e. The agreement propagation property ensures that the range of values will increase by at most e in the second stage. Therefore, the voting results for process P2 at the trustworthy LEFT nodes agree within 2e.

Case 2: $|Asym\_EV_{P2,j}| = 0$: Based on the agreement propagation property for a stage operation, the voting results for process P1 at trustworthy RIGHT nodes agree within $\Delta_{P0} + e$. The agreement generation property ensures that the voting results for process P2 at trustworthy LEFT nodes agree within e.

Agreement generation is an important property for the ROBUS clock synchronization protocols. It implies that the maximum range of the voting results for process P2 at trustworthy LEFT nodes is independent of the initial range of values transmitted by process P0.

A.6.2.2. Agreement propagation phase

An agreement propagation phase operates under the same assumptions and has the same properties as an agreement generation phase. An agreement propagation phase serves to ensure that all of the trustworthy nodes agree on the result of the agreement generation phase and to provide a way for good recovering nodes to acquire the protocol result, even if their set of eligible voters do not completely agree with the set of eligible voters at the trustworthy nodes of the same kind.

The value transmitted by process P2 for the agreement propagation phase is equal to the result of its vote for the agreement generation phase. Let $v_{P2,min}$ and $v_{P2,max}$ denote the bounds for the values transmitted by process P2 at the trustworthy LEFT nodes. The value transmitted by process P3 is equal to the result of its vote. Thus, $v_{P3,min}$ and $v_{P3,max}$ denote the bounds for the voting results and the transmitted values for process P3 at the trustworthy RIGHT nodes. $v_{P4,min}$ and $v_{P4,max}$ denote the bounds for the voting results for process P4 at the trustworthy LEFT nodes. Δ_{P2}, Δ_{P3}, and Δ_{P4} denote the bounds on the range of the voting results at the trustworthy nodes for processes P2, P3, and P4, respectively.

In general, for the trustworthy LEFT nodes, there is little or no difference between taking the voting result of P2 or P4 as the protocol result. For the actual ROBUS protocols, symmetry of implementation, timing, and others factor are taken into consideration to determine which result to use.

Good recovering nodes perform the same voting operations as trustworthy nodes of the same kind in order to acquire the result of the protocol. The processes for good recovering nodes trying to capture the protocol result are labeled P3C and P4C, corresponding to processes P3 and P4 for trustworthy nodes, respectively. The most significant difference between good recovering nodes and trustworthy nodes is that their set of eligible voters can differ in the number of trustworthy, symmetric, and asymmetric voters. For each good recovering node, it is assumed that the set of eligible voters contains more trustworthy sources than untrustworthy ones. That is, for good recovering RIGHT node i:

$$|Twy\_EV_{P3C,i}| > |Sym\_EV_{P3C,i}| + |Asym\_EV_{P3C,i}|.$$

For good recovering LEFT node j:

$$|Twy\_EV_{P4C,j}| > |Sym\_EV_{P4C,j}| + |Asym\_EV_{P4C,j}|.$$

It is not assumed that all of the trustworthy sources of the opposite kind are considered eligible voters at good recovering nodes. Since the eligible voter sets at trustworthy nodes of a particular kind are assumed to include all of the trustworthy sources of the opposite kind, the set of trustworthy eligible voters at good recovering nodes must necessarily be a subset of the set of trustworthy eligible voters at trustworthy nodes of the same kind. Thus, for good recovering RIGHT node i:

$$|Twy\_EV_{P3C,i}| \leq |Twy\_EV_{P3,i}|.$$

For good recovering LEFT node j:

$$|Twy\_EV_{P4C,j}| \leq |Twy\_EV_{P4,j}|.$$

No assumption is made about the number of asymmetric untrustworthy sources in the eligible voter sets of good recovering nodes.

The following presents properties of operations with the exact and inexact communication models for

trustworthy nodes and good recovering nodes.

A.6.2.2.1. Voting with exact communication

The following properties hold for an agreement propagation phase with exact communication.

Validity at the trustworthy RIGHT nodes: At the trustworthy RIGHT nodes, the result of the vote in process P3 is in the interval $[v_{P0,min}, v_{P0,max}]$.

Proof: Based on the validity property for a stage operation with exact communication, the voting results for process P3 at trustworthy RIGHT nodes are in the interval $[v_{P2,min}, v_{P2,max}]$. According to the validity property for the agreement generation phase, this interval is equal to $[v_{P0,min}, v_{P0,max}]$.

Validity at the good recovering RIGHT nodes: At the good recovering RIGHT nodes, the result of the vote in process P3C is in the interval $[v_{P0,min}, v_{P0,max}]$.

Proof: The proof for process P3 at the trustworthy RIGHT nodes applies here.

Agreement propagation at the trustworthy RIGHT nodes: The result of the vote for process P3 at the trustworthy RIGHT nodes is equal to the result of the vote for process P2 at the trustworthy LEFT nodes.

Proof: The agreement generation phase ensures exact agreement for process P2 at the trustworthy LEFT nodes (i.e., $v_{P2,min} = v_{P2,max}$). The validity property and the agreement propagation property for a stage operation ensures that the result for process P3 at the trustworthy RIGHT nodes exactly matches the results for process P2 at the trustworthy LEFT nodes. Thus: $v_{P3,min} = v_{P3,max} = v_{P2,min} = v_{P2,max}$.

Agreement propagation at the good recovering RIGHT nodes: The result of the vote for process P3C at the good recovering RIGHT nodes is the same as the result of the vote for process P2 at the trustworthy LEFT nodes.

Proof: The proof for process P3 at the trustworthy RIGHT nodes applies here.

Validity at the trustworthy LEFT nodes: At the trustworthy LEFT nodes, the result of the vote in process P4 is in the interval $[v_{P0,min}, v_{P0,max}]$.

Proof: Based on the validity property for a stage operation with exact communication, the voting results for process P4 at trustworthy LEFT nodes are in the interval $[v_{P3,min}, v_{P3,max}]$. According to the validity property for the trustworthy RIGHT nodes in an agreement propagation phase, this interval is equal to $[v_{P0,min}, v_{P0,max}]$.

Validity at the good recovering LEFT nodes: At the good recovering LEFT nodes, the result of the vote in process P4C is in the interval $[v_{P0,min}, v_{P0,max}]$.

Proof: The proof for process P4 at the trustworthy LEFT nodes applies here.

Agreement propagation at the trustworthy LEFT nodes: The result of the vote for process P4 at the trustworthy LEFT nodes is equal to the result of the vote for process P2 at the trustworthy LEFT nodes.

Proof: The vote results for process P3 at the trustworthy RIGHT nodes are known to exactly agree with one another and with the vote results for process P2 at the trustworthy LEFT nodes. The validity property and the agreement propagation property for a stage operation ensures that the results for process P4 at the trustworthy LEFT nodes exactly match the results for process P3 at the trustworthy RIGHT nodes. Thus: $v_{P4,min} = v_{P4,max} = v_{P2,min} = v_{P2,max}$.

Agreement propagation at the good recovering LEFT nodes: The result of the vote for process P4C at the good recovering LEFT nodes is equal to the result of the vote for process P2 at the trustworthy LEFT nodes.

Proof: The proof for process P4 at the trustworthy LEFT nodes applies here.

A.6.2.2.2. Voting with inexact communication

The following properties hold for an agreement propagation phase with inexact communication.

Validity at the trustworthy RIGHT nodes: At the trustworthy RIGHT nodes, the result of the vote in process P3 is in the interval $[v_{P0,min} - 3\varepsilon_l, v_{P0,max} - 3\varepsilon_h]$.

Proof: The application of the validity property for stages 1 through 3 constrains the result of voting operations in process P3 at the trustworthy RIGHT nodes to the interval $[v_{P0,min} - 3\varepsilon_l, v_{P0,max} - 3\varepsilon_h]$.

Validity at the good recovering RIGHT nodes: At the good recovering RIGHT nodes, the result of the vote in process P3C is in the interval $[v_{P0,min} - 3\varepsilon_l, v_{P0,max} - 3\varepsilon_h]$.

Proof: The proof for process P3 at the trustworthy RIGHT nodes applies here.

Agreement at the trustworthy RIGHT nodes: The voting results for process P3 at the trustworthy RIGHT nodes agree within 2e (i.e., $\Delta_{P3} = 2e$).

Proof: Two cases must be considered.

Case 1: $|Asym\_EV_{P3,i}| = 0$: According to the agreement generation property for a stage operation with inexact communication, the voting results for process P3 at the trustworthy RIGHT nodes will agree within e.

Case 2: $|Asym\_EV_{P2,j}| = 0$: The agreement generation property for a stage operation with inexact communication ensures that the voting results for process P2 at trustworthy LEFT nodes agree within e. The agreement propagation property for a stage operation ensures that the voting results for process P3 at trustworthy RIGHT nodes differ by at most another e.

Agreement at the good recovering RIGHT nodes: The voting results for process P3C at the good recovering RIGHT nodes agree within 3e (i.e., $\Delta_{P3C} = 3e$).

Proof: The number of asymmetric untrustworthy eligible voters for process P3C may be nonzero. The worst case agreement among the voting results for process P2 is 2e (i.e., $\Delta_{P2} = 2e$). The agreement propagation property for a stage operation ensures that the voting results for process P3C at good recovering RIGHT nodes differ by at most another e. Thus: $\Delta_{P3C} = \Delta_{P2} + e = 3e$.

Agreement propagation at the trustworthy RIGHT nodes: The voting results for process P3 at the trustworthy RIGHT nodes and the voting results for process P2 at the trustworthy LEFT nodes agree within $\Delta_{P2} + \max(\varepsilon_l, \varepsilon_h)$.

Proof: The voting results for process P2 at the trustworthy LEFT nodes are in the interval $[v_{P2,min}, v_{P2,max}]$ and agree within Δ_{P2}. The voting results for process P3 at the trustworthy RIGHT nodes are in the interval $[v_{P2,min} - \varepsilon_l, v_{P2,max} + \varepsilon_h]$. Thus, the maximum difference between voting results at P2 and P3 is $\max((v_{P2,max} + \varepsilon_h) - v_{P2,min}, v_{P2,max} - (v_{P2,min} - \varepsilon_l)) = \Delta_{P2} + \max(\varepsilon_l, \varepsilon_h)$.

Agreement propagation at the good recovering RIGHT nodes: The voting results for process P3C at the good recovering RIGHT nodes and the voting results for process P2 at the trustworthy LEFT nodes agree within $\Delta_{P2} + \max(\varepsilon_l, \varepsilon_h)$.

Proof: The proof for process P3 at the trustworthy RIGHT nodes applies here.

Agreement between trustworthy RIGHT nodes and good recovering RIGHT nodes: The voting results for processes P3 at the trustworthy RIGHT nodes and the voting results for P3C at the good recovering RIGHT nodes agree within 3e.

Proof: The worst case agreement among the voting results for process P2 is 2e (i.e., $\Delta_{P2} = 2e$). According to the validity property for a stage operation, the voting results for processes P3 at the trustworthy RIGHT nodes and P3C at the good recovering RIGHT nodes are in the interval $[v_{P2,min} - \varepsilon_l, v_{P2max} - \varepsilon_h]$, which has a range of 3e. The voting results for process P3 at the trustworthy RIGHT nodes are in the validity interval and agree with one another within 2e. However, although the voting results for process P3C at the good recovering RIGHT nodes are in the validity interval, their agreement bound is 3e, which is equal to the range of the validity interval. In the worst case, voting results for processes P3 and P3C can be at opposite extremes of the validity interval and differ by at most 3e.

Validity at the trustworthy LEFT nodes: At the trustworthy LEFT nodes, the result of the vote in process P4 is in the interval $[v_{P0,min} - 4\varepsilon_l, v_{P0,max} - 4\varepsilon_h]$.

Proof: The application of the validity property for stages 1 through 4 constrains the result of voting operations in process P4 at the trustworthy LEFT nodes to the interval $[v_{P0,min} - 4\varepsilon_l, v_{P0,max} - 4\varepsilon_h]$.

Validity at the good recovering LEFT nodes: At the good recovering LEFT nodes, the result of the vote in process P4C is in the interval $[v_{P0,min} - 4\varepsilon_l, v_{P0,max} - 4\varepsilon_h]$.

Proof: The proof for process P4 at the trustworthy LEFT nodes applies here.

Agreement at the trustworthy LEFT nodes: The voting results for process P4 at the trustworthy LEFT nodes agree within 2e (i.e., $\Delta_{P4} = 2e$).

Proof: Two cases must be considered.

Case 1: $|Asym\_EV_{P3,i}| = 0$: According to the agreement generation property for a stage operation with inexact communication, the voting results for process P3 at the trustworthy RIGHT nodes will agree within e. The agreement propagation property for a stage operation ensures that the voting results for process P4 at trustworthy LEFT nodes differ by at most another e.

Case 2: $|Asym\_EV_{P4,j}| = 0$: The agreement generation property for a stage operation with inexact communication ensures that the voting results for process P4 at trustworthy LEFT nodes agree within e.

Agreement at the good recovering LEFT nodes: The voting results for process P4C at the good recovering LEFT nodes agree within 3e (i.e., $\Delta_{P4C} = 3e$).

Proof: The number of asymmetric untrustworthy eligible voters for process P4C may be nonzero. The worst case agreement among the voting results for process P3 is 2e (i.e., $\Delta_{P3} = 2e$). The agreement propagation property for a stage operation ensures that the voting results for process P4C at good recovering LEFT nodes differ by at most another e.

Agreement propagation at the trustworthy LEFT nodes: The voting results for process P4 at the trustworthy LEFT nodes and the voting results for process P2 at the trustworthy LEFT nodes agree within $\Delta_{P2} + 2*\max(\varepsilon_l, \varepsilon_h)$.

Proof: The voting results for process P2 at the trustworthy LEFT nodes are in the interval $[v_{P2,min}, v_{P2,max}]$ and agree within Δ_{P2}. Application of the validity property for stages 3 and 4 constrains the voting results for process P4 at the trustworthy RIGHT nodes to the interval $[v_{P2,min} - 2\varepsilon_l, v_{P2,max} + 2\varepsilon_h]$. Thus, the maximum difference between voting results at P2 and P4 is $\max((v_{P2,max} + 2\varepsilon_h) - v_{P2,min}, v_{P2,max} - (v_{P2,min} - 2\varepsilon_l)) = \Delta_{P2} + 2*\max(\varepsilon_l, \varepsilon_h)$.

Agreement propagation at the good recovering LEFT nodes: The voting results for process P4C at the good recovering LEFT nodes and the voting results for process P2 at the trustworthy LEFT nodes agree within $\Delta_{P2} + 2*\max(\varepsilon_l, \varepsilon_h)$.

Proof: The proof for process P4 at the trustworthy LEFT nodes applies here.

Agreement between trustworthy LEFT nodes and good recovering LEFT nodes: The voting results for processes P4 at the trustworthy LEFT nodes and the voting results for P4C at the good recovering LEFT nodes agree within 3e.

Proof: The worst case agreement among the voting results for process P3 is 2e (i.e., $\Delta_{P3} = 2e$). According to the validity property for a stage operation, the voting results for processes P4 at the trustworthy LEFT nodes and P4C at the good recovering LEFT nodes are in the interval $[v_{P3,min} - \varepsilon_l, v_{P3max} - \varepsilon_h]$, which has a range of 3e. The voting results for process P4 at the trustworthy LEFT nodes are in the validity interval and agree with one another within 2e. Although the voting results for process P4C at the good recovering LEFT nodes are in the validity interval, their agreement is within 3e, which is equal to the range bound of the validity interval. In the worst case, voting results for processes P4 and P4C can be at opposite extremes of the validity interval and differ by at most 3e.

A.7. Stage operations of ROBUS protocols

The ROBUS protocols process PE messages and state data. The required computation for the actual protocols goes beyond the basic middle-value-select voting function. However, the dynamic middle-value-select voting function is adaptable enough to cover all the required stage operations that involve computation processes at the ROBUS nodes. The following subsections describe how a middle-value-select voter can be adapted for the actual ROBUS protocols. The main purpose of this section is to

establish a link between the theory presented in this appendix and the actual ROBUS protocols.

A.7.1. Event voting

The time synchronization protocols use middle-value-select voting for the processing of timing events. The voting function for these protocols is referred to as the **Accept** function. These protocols use a fixed-delay communication model, which corresponds to the inexact communication model presented above in terms of the precision of received values.

A.7.2. Routing

A ROBUS node performs a routing function by relaying a message received from a particular input source. A middle-value-select voter can perform a routing function by including in the set of eligible voters only the input source of interest. This function is used with the synchronous communication model, which corresponds to the exact communication model in the protocol theory. As implemented in the ROBUS protocols, voter eligibility for the routing function also takes into consideration input errors and local accusations. The special case of an empty set of eligible voters is handled to ensure protocol results consistent with the basic protocol theory presented in the previous section.

A.7.3. Word voting

The unit of data for word voting is the ROBUS Message. The word voting function implemented in the ROBUS protocols is an exact-match majority word vote. This function is used with the synchronous communication model, which corresponds to the exact communication model. The vote result equals the majority value if an exact-match majority exists (i.e., a majority of eligible inputs are exactly equal). Otherwise, the result is invalid and a signal is asserted indicating that there is not a majority value among the eligible inputs.

If there is exact agreement among a majority of the values received from eligible voters, then the result of a word vote equals the result of a middle-value-select vote with the same set of eligible voters. The ROBUS protocols with word voting handle a no-majority condition as an exception, and the corresponding result and interpretation depends on the protocol and the protocol stage being executed.

A.7.4. Bit voting

The data for bit voting are the bits from the Payload field of ROBUS Messages. Bit voting is used to process diagnostic data. This function is used with the synchronous communication model, which corresponds to the exact communication model. The bits are interpreted as Boolean variables with TRUE or FALSE values. Bit voting is an exact-match majority bit vote in which the result of a vote equals the value of the majority if an exact-match majority exists. Otherwise, the result is equal to TRUE.

As for word voting, if there is exact agreement among a majority of the values received from eligible voters, then the result of a bit vote equals the result of a middle-value-select vote with the same set of eligible voters. The no-majority condition is an exceptional case compatible in terms of validity and agreement with the protocol theory presented previously.

A.8. ROBUS fault assumptions

The ROBUS fault assumptions are derived from the generic analysis presented in this appendix and the specific protocol analyses presented in other appendices of this document. The assumptions are sufficient conditions for ensuring that the protocol results are correct. The assumptions depend on the mode of operation.

A.8.1.1. Clique Initialization and Clique Preservation modes

The following conditions are assessed for each voting protocol executed in these modes. A violation of these conditions may result in a protocol failure. Failure is assessed from the perspective of a clique rather than individual nodes.

- For each receiving process at each trustworthy node, all trustworthy sources of the opposite kind are eligible to vote.

- For each receiving process, the trustworthy receiving nodes of a given kind agree on the eligibility of non-asymmetric sources of the opposite kind.

- There are no asymmetric eligible voters for any of the receiving processes at every trustworthy BIU receiver or at every trustworthy RMU receiver.

- For each receiving process at each trustworthy node, the set of eligible voters contains more trustworthy sources than untrustworthy ones. (For the Schedule Update and the PE Broadcast protocols, the number of eligible voters for process P1 may be zero without compromising the protocol properties. This is examined in Appendices D and E, respectively.)

A.8.1.2. Clique Join mode

The following conditions apply to good recovering nodes in the Clique Join mode. The conditions are assessed for each voting protocol executed in this mode. A violation of these conditions may result in a protocol failure for the recovering node.

- For each receiving process, all trustworthy sources of the opposite kind are eligible to vote.

- For each receiving process, the recovering node agrees with the trustworthy nodes of the same kind on the eligibility of non-asymmetric sources of the opposite kind.

- There are no asymmetric eligible voters for any of the receiving processes at the good recovering node and every trustworthy receiver of the same kind, or at every trustworthy receiver of the opposite kind.

- For each receiving process, the set of eligible voters contains more trustworthy sources than untrustworthy ones. (For the Schedule Update and the PE Broadcast protocols, the number of eligible voters for process P1 may be zero without compromising the protocol properties. This is examined in Appendices D and E, respectively.)

A.8.1.3. Clique Detection mode

The following conditions apply to good recovering nodes in the Clique Detection mode. The conditions are assessed for each voting protocol executed in this mode. A violation of these conditions may result in a protocol failure for the recovering node.

- For each receiving process, the set of eligible voters contains more trustworthy sources than untrustworthy ones.

Appendix B. Point-to-point communication

This appendix examines the point-to-point communication between ROBUS nodes. Each ROBUS node is driven by an independent physical oscillator and a **logical time clock**, referred to as a **local-time clock**, that keeps track of the passage of time as indicated by the oscillator. The Communication Module of each ROBUS node is composed of transmit and receive sub-modules. The transmit sub-module consists of one or more separate transmitters to support broadcast transmissions. The receive sub-module consists of a separate receiver for each node of the opposite kind. The transmitters and receivers are expected to be generic components supporting event-triggered communication. The granularity of a Communication Module transaction should be a ROBUS Message, since the communication, processing, and diagnosis performed by the ROBUS protocols are based on single-message transactions. For the transmitters, the reading of a new message and the beginning of its transmission process is triggered by a send signal at the transmitter's input interface. Similarly, the receivers should be able to receive new messages whenever they arrive. The only expected communication throughput constraint at the input interface of the transmitters is the minimum data introduction interval (DII), which is the minimum number of clock ticks between consecutive requests to send messages.

The communication system must be able to support the fixed-delay and synchronous communication models. For some receiver designs, the output signals from the receiver are not synchronized to the circuitry-driving signal generated by a local physical oscillator. Therefore, the Computation Module must synchronize each received message with respect to the local oscillator before proceeding with further processing. For the synchronous communication model, the processing of received messages is triggered by the local-time clock. Therefore, a node must be able to buffer received messages until it is time to process them. The timing design of the system must be able to handle the uncertainty in the time of transmission, the transmission delay, and the synchronization delay.

In addition, this version of the ROBUS is intended to demonstrate that the bus can achieve a PE-message throughput that approaches the available bandwidth at the physical links. For most transmissions, it is possible to compute a local-time interval during which a receiver should expect to receive the message. For low link data rates, the reception intervals for individual nodes do not overlap and each message can be processed before the next one arrives. For high data rates, the reception intervals of consecutive messages overlap and the processing must be pipelined in order to match the link throughput. This appendix examines some critical aspects of pipelined communication.

In what follows, the term **oscillator clock** denotes the signal generated by the physical oscillator, the **local-time clock** refers to the logical-time clock, and the **local time** refers to the state of the logical-time clock. The process of synchronizing a signal or a message to the transitions of the oscillator clock is referred to as **signal synchronization**. The process of synchronizing a message to the local time is referred to as **deskewing**.

B.1. Physical oscillators and local-time clocks

Each ROBUS node is driven by an independent, free-running physical oscillator (i.e., the phase is not controlled in any way) and a logical-time clock (i.e., a counter) that keeps track of the passage of time as indicated by the oscillator. An oscillator **tick**, also called a **clock tick** or a

system tick, is the basic unit of time on the bus. Let f_0 denote the nominal frequency of an oscillator measured in ticks per second or Hertz (Hz). The duration of a tick for an ideal oscillator is exactly $1/f_0$ seconds. An ideal oscillator is said to have zero drift rate with respect to real-time since the oscillator perfectly marks the passage of time with a tick duration of exactly $1/f_0$ seconds. Real oscillators are characterized by non-zero drift rates with respect to real-time. It is assumed that the drift rate of the physical oscillators is bounded by a small positive constant ρ_0, which is positive, real valued, and unitless.

The bound on the drift of the physical oscillators is interpreted as follows. Let $c_x(T)$ denote the earliest real time at which local-time clock x reaches value T. $c_x(T)$ has units of **nominal ticks** (1 nominal tick = $1/f_0$ seconds). T_1 and T_2 denote arbitrary values of the local-time clock with the constraint $T_2 \geq T_1$. Then:

$$(T_2 - T_1)/(1 + \rho_0) \leq c_x(T_2) - c_x(T_1) \leq (1 + \rho_0)(T_2 - T_1) \qquad \text{(B.1)}$$

Let τ_0 denote the nominal tick duration measured in seconds (i.e., 1 nominal tick = τ_0 seconds = $1/f_0$ seconds). τ_x denotes the actual tick duration of local-time clock x. The bound on the drift rate of clock x can be expressed as follows:

$$\tau_0/(1 + \rho_0) \leq \tau_x \leq (1 + \rho_0)\tau_0 \qquad \text{(B.2)}$$

In other words, the fastest clock has a tick duration of at least $1/(1 + \rho_0)$ nominal ticks, and the slowest clock has a tick duration of at most $(1 + \rho_0)$ nominal ticks. This simple model accounts for the drift with respect to real time of the physical oscillators and the local-time clocks. The point-to-point communication model accounts for jitter on the output of the physical oscillator.

B.2. Synchronization of asynchronous signals

Single-phase edge-triggered flip-flops used as building blocks in traditional synchronous sequential digital circuits have a simple nominal timing behavior: If the signal at the data input is stable within a specified window around the oscillator clock's triggering edge, then the input value will propagate to the output of the flip-flop and stabilize within some guaranteed time. The propagation delay of the flip-flops is the time elapsed from the triggering edge of the oscillator clock until the output is stable. The window around the oscillator clock's triggering edge is characterized by the setup and hold time of the flip-flop. The **setup time** is the minimum time that the input signal must remain stable before the triggering edge of the oscillator clock in order for the output of the flip-flop to meet the nominal propagation delay. The **hold time** is the minimum time that the input signal must remain stable after the triggering edge of the oscillator clock in order for the output of the flip-flop to meet the nominal propagation delay.

The **domain** of an oscillator clock includes all the digital circuitry driven by that signal. A signal is said to be **synchronous** with respect to a particular oscillator clock if the timing of the signal meets the input setup and hold time constraints of the flip-flops driven the oscillator clock. A signal that does not meet these constraints is called **asynchronous** with respect to the given oscillator clock. Since the oscillator clocks in the fault containment regions of the ROBUS are independent and the timing of their transitions is not coordinated in any way, any signal crossing from one FCR to another is considered asynchronous when it arrives at the receiving FCR.

Asynchronous signals must be synchronized to the oscillator clock before they can be processed. Various mechanisms can be used to achieve this synchronization. Ultimately, however, consideration must be given to the problem of violations of the setup and hold times of flip-flops reading the signal. A flip-flop sampling an input that is not stable within the setup and hold window can enter a metastable condition in which the output does not settle to a valid logic state within the nominal propagation delay. If not handled properly, this can result in the generation of more asynchronous signals and the propagation of errors throughout the receiving FCR.

The mean time between failure (MTBF) for a flip-flop reading an asynchronous input is (see [XAPP077]):

$$MTBF = e^{-C_2 * t_{MET}} / (2 * C_1 * f_D * f_C) \qquad (B.3)$$

where t_{MET} denotes the time available for the metastability to resolve itself (i.e., time allowed by downstream circuitry before reading the output of the flip-flop), f_D denotes the input signal frequency ($2*f_D$ is the input signal event rate), f_C denotes the oscillator clock frequency, C_1 denotes the metastability aperture of the flip-flop (related to the width of the window during which an input can cause a metastability condition), and C2 denotes the resolution rate (related to the speed with which the metastable condition will be resolved). Constants C1 and C2 are functions of the process technology and flip-flop design. For current technology, the variables of the MTBF can be selected such that the probability of metastability failures is extremely small.

In what follows, it is assumed that the problem of metastability is properly handled by the implementation of the ROBUS. For analysis, unless explicitly stated otherwise, it is assumed that the nodes have ideal signal synchronizers, each consisting of a single flip-flop driven by the oscillator clock. These ideal flip-flops have no metastable states and zero propagation delay. The timing behavior is as follows.

If the input changes just before the triggering-edge of the oscillator clock, this latest input value will propagate to the output as soon as the triggering-edge of the oscillator clock arrives. If the input changes at exactly the same time as the triggering-edge of the oscillator clock, the input value will not affect the output until the next triggering-edge of the oscillator clock (assuming that the input remains constant).

B.3. Single-message communication

The communication of a message from a source node to a receiver node is modeled as a four step process: (1) **Send**: The Computation Module of the source node signals the transmitter(s) in the Communication Module that a message is ready for transmission; (2) **Transmission**: The transmitter reads the message and transmits the corresponding signals over the transmission medium; (3) **Delivery**: The link receiver gets the message from the transmission medium and signals the arrival to the signal synchronizer; (4) **Reception**: The synchronizer signals the arrival of a new message to the Computation Module. Figure B.1 illustrates the point-to-point communication path. CLK_{Rx} denotes the oscillator clock at the receiving node. The **message delivery delay** is the time elapsed from the instant a transmitter receives a send request until the message is presented at the output interface of the receiver. The **message reception delay** is

equal to the message delivery delay plus the additional time delay to synchronize the received message to the oscillator clock at the receiving node.

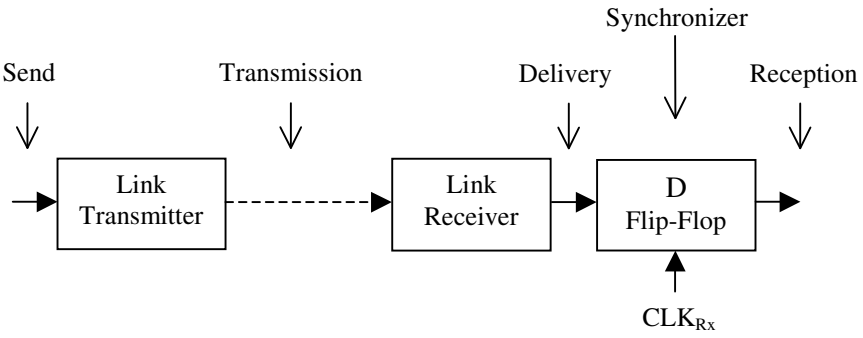


Figure B.1: Point-to-point communication path

Let $d_{PP,l}$ and $d_{PP,h}$ denote the minimum and maximum point-to-point message delivery delays, respectively, measured in units of nominal clock ticks. v_{PP} denotes the delivery precision (i.e., the uncertainty in the point-to-point delivery delay) measured in units of nominal clock ticks. $r_{PP,l}$ and $r_{PP,h}$ denote the minimum and maximum point-to-point message reception delays, respectively, measured in nominal clock ticks. e_{PP} denotes the reception precision (i.e., the uncertainty in the point-to-point reception delay) measured in nominal clock ticks.

B.3.1. Reception delay

Let T_0 denote the local time at which the source sends the message, and let t_0 denote the corresponding real time. The real-time range of point-to-point message delivery is $[t_0 + d_{PP,l}, t_0 + d_{PP,h}]$. Therefore, the delivery precision is:

$$v_{PP} = d_{PP,h} - d_{PP,l} \quad \text{(B.4)}$$

The minimum point-to-point message reception delay happens when the message is sampled by the input synchronizer at exactly the same time it is delivered.

$$r_{PP,l} = d_{PP,l} \quad \text{(B.5)}$$

The maximum point-to-point message reception delay happens when the message is sampled by the input synchronizer exactly one tick after it is delivered. The worst case delay occurs when the oscillator clock at the receiving node is slow.

$$r_{PP,h} = d_{PP,h} + (1 + \rho_0) \quad \text{(B.6)}$$

The real-time range of reception is $[t_0 + r_{PP,l}, t_0 + r_{PP,h}]$. Therefore, the reception precision is:

$$e_{PP} = r_{PP,h} - r_{PP,l} = [d_{PP,h} + (1 + \rho_0)] - d_{PP,l} = 1 + \rho_0 + v_{PP} \quad \text{(B.7)}$$

e_{PP} accounts for time-discretization errors, jitter and drift of the source and recover oscillators, and slight differences in point-to-point communication delay due to different length wires/fibers.

Next, we define $IMP(x_1, x_2)$, the Integer Mid-Point value, as the integer closest to the mid-

point of x_1 and x_2. $IMP(x_1, x_2)$ is computed in two steps:

Step 1: $x = (x_1 + x_2)/2$

Step 2: IMP = round(x), with $round(x) = \lfloor x \rfloor$ if $x < ½$, or $\lceil x \rceil$ if $x \geq ½$

R_{PP} denotes the expected reception delay:

$$R_{PP} = IMP(r_{PP,l}, r_{PP,h}) \quad \text{(B.8)}$$

B.3.2. Estimate of the local-time at the source

Let T_{RCV} denote the local time at the receiver when it receives the message. To estimate the local time at the source node, the receiver assumes that the message reception delay is R_{PP} ticks of its oscillator clock. The estimated local time at the source node at the time of reception is:

$$T_{SRC,E} = T_0 + R_{PP} \quad \text{(B.9)}$$

The error in the local-time estimate is bounded as follows. T_{RCV} occurs no earlier than $\mu_{PP,l}$ nominal ticks from the actual local time $T_{SRC,E}$ at the source:

$$\mu_{PP,l} = (1 + \rho_0)R_{PP} - r_{PP,l} \quad \text{(B.10)}$$

T_{RCV} occurs no later than $\mu_{PP,h}$ nominal ticks from the actual local time $T_{SRC,E}$ at the source:

$$\mu_{PP,h} = r_{PP,h} - R_{PP}/(1 + \rho_0) \quad \text{(B.11)}$$

B.3.3. Expected local time of reception

Let $\pi_{PP,SR}$ denote a bound on the relative local-time skew between the source and the receiver nodes. This bound is assumed to hold for the duration of the communication. The expected local time of reception at the receiver is denoted by $T_{RCV,E}$.

$$T_{RCV,E} = T_0 + R_{PP} \quad \text{(B.12)}$$

Due to the relative local-time skew and the uncertainty in the message-reception delay, the message will arrive within some local-time interval containing $T_{RCV,E}$. Let $\Delta_{PP,RCV}$ denote the local-time error in T_{RCV}:

$$\Delta_{PP,RCV} = T_{RCV} - T_{RCV,E} \quad \text{(B.13)}$$

We want to determine the absolute maximum local-time error in T_{RCV}, denoted by $\Delta_{PP,RCV}|_{abs\text{-}max}$:

$$|T_{RCV} - T_{RCV,E}| \leq \Delta_{PP,RCV}|_{abs\text{-}max} \quad \text{(B.14)}$$

The value of $\Delta_{PP,RCV}|_{abs\text{-}max}$ is derived as follows. The bound on the local-time synchronization between the source and the receiver nodes is expressed as:

$$|c_{SRC}(T) - c_{RCV}(T)| \leq \pi_{PP,SR} \quad \text{(B.15)}$$

where $c_{SRC}(T)$ and $c_{RCV}(T)$ denote the earliest real times at which the local times at the source and at the receiver, respectively, reach value T. From the previous analysis, it is known that the real-time difference between the time when the source reaches $T_{RCV,E}$ and the time when the message is actually received T_{RCV} is bounded above and below by $\mu_{PP,h}$ and $\mu_{PP,l}$, respectively.

$$c_{SRC}(T_{RCV,E}) - \mu_{PP,l} \leq c_{RCV}(T_{RCV}) \leq c_{SRC}(T_{RCV,E}) + \mu_{PP,h} \quad \text{(B.16)}$$

For local time $T_{RCV,E}$, inequality (B.15) can be re-expressed as:

$$c_{SRC}(T_{RCV,E}) - \pi_{PP,SR} \leq c_{RCV}(T_{RCV,E}) \leq c_{SRC}(T_{RCV,E}) + \pi_{PP,SR} \quad \text{(B.17)}$$

Combining inequalities (B.16) and (B.17), we get:

$$-\pi_{PP,SR} - \mu_{PP,l} \leq c_{RCV}(T_{RCV}) - c_{RCV}(T_{RCV,E}) \leq \pi_{PP,SR} + \mu_{PP,h} \quad \text{(B.18)}$$

So:

$$|c_{RCV}(T_{RCV}) - c_{RCV}(T_{RCV,E})| \leq \max(\pi_{PP,SR} + \mu_{PP,l}\ ,\ \pi_{PP,SR} + \mu_{PP,h}) \quad \text{(B.19)}$$

Equivalently:

$$|c_{RCV}(T_{RCV}) - c_{RCV}(T_{RCV,E})| \leq \pi_{PP,SR} + \max(\mu_{PP,l}\ ,\ \mu_{PP,h}) \quad \text{(B.20)}$$

Using the constraint that the local clocks are ρ-bounded, the definition of $\Delta_{PP,RCV}$, and the real time duration of $\Delta_{PP,RCV}$ ticks for the fastest allowed clock:

$$|\Delta_{PP,RCV}|/(1 + \rho_0) \leq |c_{RCV}(T_{RCV}) - c_{RCV}(T_{RCV,E})| \quad \text{(B.21)}$$

Combining (B.20) and (B.21):

$$|\Delta_{PP,RCV}| \leq (1 + \rho_0)(\pi_{PP,SR} + \max(\mu_{PP,l}\ ,\ \mu_{PP,h})) \quad \text{(B.22)}$$

Since $\Delta_{PP,RCV}$ is an integer, we can take the floor in (B.22):

$$|\Delta_{PP,RCV}| \leq \lfloor (1 + \rho_0)(\pi_{PP,SR} + \max(\mu_{PP,l}\ ,\ \mu_{PP,h})) \rfloor \quad \text{(B.23)}$$

Therefore, the worst-case local-time difference between the actual time of reception T_{RCV} and the expected time of reception $T_{RCV,E}$ is:

$$\Delta_{PP,RCV}|_{abs\text{-}max} = \lfloor (1 + \rho_0)(\pi_{PP,SR} + \max(\mu_{PP,l}\ ,\ \mu_{PP,h})) \rfloor \quad \text{(B.24)}$$

B.4. Coordination for synchronous communication

For the synchronous ROBUS protocols, the scheduling of operations is based on a **distributed synchronous composition** abstract model of the system in which a single oscillator drives a common local-time clock and fixed-delay processes corresponding to the communication and

computation operations of the BIUs and RMUs. Communication during time-driven operations is time-triggered. For each transmission, the sources and receivers use a particular local-time value as a distributed reference event to coordinate their actions. Given specific bounds for the reception delay and the relative local-time skew between sources and receivers, it is possible to coordinate the send and receive operations such that the transmitted messages are received within a predetermined local-time range measured at the receivers. The receivers can then apply a deskewing function and forward the received messages for processing at a predetermined local time. By leveraging the previous analysis, it is possible to analyze the source-receiver coordination problem using only global time (i.e., synchronized local time viewed from a global perspective). Figure B.2 illustrates the relevant timing events. T_{REF} denotes the reference local-time value. T_{SND} is the time at which the message is sent. R_{PP} is the expected reception delay. $T_{RCV,E}$ is the expected time of reception. W_{Deskew} is the size of the deskewing window. $W_{Deskew,pre}$ is the pre-expectation window (i.e., the size of the section of the deskewing window before the expected time of reception). $W_{Deskew,post}$ is the post-expectation window (i.e., the size of the deskewing window after the expected time of reception). $T_{PROC,begin}$ denotes the time for the beginning of message processing.

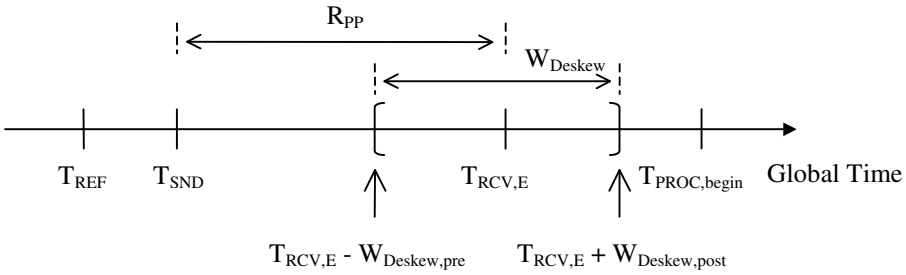


Figure B.2: Timing events for point-to-point communication

A message from a good source is expected to arrive during the following closed time interval, which includes all triggering edges of the local clock within the expected time range of reception:

$$[T_{RCV,E} - \Delta_{PP,RCV}|_{abs\text{-}max}, T_{RCV,E} + \Delta_{PP,RCV}|_{abs\text{-}max}] \quad \text{(B.25)}$$

The deskewing window includes all triggering edges of the clock within the expected time range of reception, a total of $2\Delta_{PP,RCV}|_{abs\text{-}max} + 1$ edges. The deskewing window is intended to cover the duration of all local clock counts corresponding to the triggering edges of the clock within the expected time range of reception. The local clock counts corresponding to these triggering edges determine a time interval with a duration of $2\Delta_{PP,RCV}|_{abs\text{-}max} + 1$ ticks. The deskewing window extends for the real-time interval corresponding to the following half-closed local-time interval:

$$[T_{RCV,E} - \Delta_{PP,RCV}|_{abs\text{-}max}, T_{RCV,E} + \Delta_{PP,RCV}|_{abs\text{-}max} + 1) \quad \text{(B.26)}$$

So:

$$W_{Deskew} = 2\Delta_{PP,RCV}|_{abs\text{-}max} + 1 \quad \text{(B.27)}$$

Then:

$$W_{Deskew,pre} = \Delta_{PP,RCV}|_{abs\text{-}max} \quad \text{(B.28)}$$

$$W_{Deskew,post} = \Delta_{PP,RCV}|_{abs\text{-}max} + 1 \quad \text{(B.29)}$$

For proper communication, the following constraints must be satisfied:

$$T_{REF} \leq T_{SND} \quad \text{(B.30)}$$

$$T_{REF} \leq T_{RCV,E} - W_{Deskew,pre} \quad \text{(B.31)}$$

$$T_{RCV,E} = T_{SND} + R_{PP} \quad \text{(B.32)}$$

$$T_{RCV,E} + W_{Deskew,post} \leq T_{PROC,begin} \quad \text{(B.33)}$$

Relations (B.30) and (B.31) express basic time constraints given by the use of a common reference time. Relation (B.32) captures the goal of source-receiver coordination, which is to receive the message at the expected time of reception. Relation (B.33) is only relevant to the composition of operations at the receiving node. Let $\Delta_{REF\text{-}SND}$ denote the delay from T_{REF} to T_{SND} measured in local clock ticks.

$$\Delta_{REF\text{-}SND} = T_{SND} - T_{REF} \quad \text{(B.34)}$$

Let $\Delta_{REF\text{-}RCVWND}$ denote the delay from T_{REF} to $T_{RCV,E} - W_{Deskew,pre}$ measured in local clock ticks.

$$\Delta_{REF\text{-}RCVWND} = (T_{RCV,E} - W_{Deskew,pre}) - T_{REF} \quad \text{(B.35)}$$

Let $\Delta_{REF\text{-}SND}|_{min}$ and $\Delta_{REF\text{-}RCVWND}|_{min}$ denote the minimum values for $\Delta_{REF\text{-}SND}$ and $\Delta_{REF\text{-}RCVWND}$, respectively. Relation (B.32) can be re-expressed as follows:

$$\Delta_{REF\text{-}SND} + R_{PP} = \Delta_{REF\text{-}RCVWND} + W_{Deskew,pre} \quad \text{(B.36)}$$

We are interested in finding the values for $\Delta_{REF\text{-}SND}$ and $\Delta_{REF\text{-}RCVWND}$ to achieve the earliest communication satisfying (B.30), (B.31), and (B.32). We consider two cases.

Case 1: $\Delta_{REF\text{-}SND}|_{min} + R_{PP} \geq \Delta_{REF\text{-}RCVWND}|_{min} + W_{Deskew,pre}$

For this case, the message can be sent as soon as possible, but the window must be delayed to align it with the expected time of reception.

$$\Delta_{REF\text{-}SND} = \Delta_{REF\text{-}SND}|_{min} \quad \text{(B.37)}$$

$$\Delta_{REF\text{-}RCVWND} = \Delta_{REF\text{-}SND}|_{min} + R_{PP} - W_{Deskew,pre} \quad \text{(B.38)}$$

Case 2: $\Delta_{REF\text{-}SND}|_{min} + R_{PP} < \Delta_{REF\text{-}RCVWND}|_{min} + W_{Deskew,pre}$

For this case, the window can be opened as soon as possible, but the message must be delayed to achieve proper alignment.

$$\Delta_{REF\text{-}SND} = \Delta_{REF\text{-}RCVWND}|_{min} + W_{Deskew,pre} - R_{PP} \quad \text{(B.39)}$$

$$\Delta_{REF\text{-}RCVWND} = \Delta_{REF\text{-}RCVWND}|_{min} \quad \text{(B.40)}$$

B.5. Message streams

Each message in a message stream is processed independently. Let K denote the total number of messages in the stream. i denotes the index for the messages in the stream, with $0 \le i \le K-1$. $T_{SND,i}$ is the local time at which the source sends the i-th message of the stream. $T_{RCV,E,i}$ is the expected local time of reception for the i-th message of the stream. Λ_{stream} denotes the data introduction interval at the source measured in local clock ticks.

The throughput capacities of the Communication Module and the Computation Module are characterized by their respective minimum data introduction interval [De Micheli 94]. Let Λ_{Comm} and Λ_{Comp} denote the minimum data introduction interval for the Communication Module and the Computation Module, respectively. Λ_{Comm} and Λ_{Comp} are measured in local-clock ticks. For proper processing, Λ_{stream} must be larger than Λ_{Comm} and Λ_{Comp}.

$$\Lambda_{stream} \ge \max(\Lambda_{Comm}, \Lambda_{Comp}) \tag{B.41}$$

B.5.1. Message delivery rate

We would like to compute the number of messages that can be delivered during a particular time interval. We consider intervals during steady state transmission after the leading edge of the stream and before the trailing edge. Because of the drift rate of the clocks, the observed number of delivered messages can vary within a range.

Let W_{RCV} denote the size of the observation window at the receiving node measured in local clock ticks. Q denotes the number of messages delivered during the observation window. λ_{SRC} denotes the data introduction interval measured in nominal clock ticks. w_{RCV} denotes the size of the observation window at the receiving node measured in nominal clock ticks. Let $t_{deliver,i}$ denote the real time at which message i is delivered.

$$t_{deliver,i} = t_{deliver,0} + i\lambda_{SRC} \tag{B.42}$$

Let $t_{obs,l}$ and $t_{obs,h}$ denote the beginning and end times, respectively, for the observation window. The observer records received messages during the closed interval $[t_{obs,l}, t_{obs,h}]$. $t_{obs,l}$ and $t_{obs,h}$ are related by the size of the observation window.

$$t_{obs,h} = t_{obs,l} + w_{RCV} \tag{B.43}$$

The following constraints are applied in order to determine the number of observed messages.

$$t_{deliver,0} < t_{obs,l} \tag{B.44}$$

$$t_{deliver,1} \ge t_{obs,l} \tag{B.45}$$

$$t_{deliver,Q} \le t_{obs,h} \tag{B.46}$$

$$t_{deliver,Q+1} > t_{obs,h} \tag{B.47}$$

For these constraints, a total of Q messages in the index range 1 to Q are delivered within the observation interval. The maximum value of Q is derived as follows. Relation (B.46) can be re-expressed as:

$$t_{deliver,0} + Q\lambda_{SRC} \leq t_{obs,1} + w_{RCV} \quad \text{(B.48)}$$

So:

$$Q \leq [(t_{obs,1} - t_{deliver,0}) + w_{RCV}]/\lambda_{SRC} \quad \text{(B.49)}$$

The right-hand side reaches its maximum value when $t_{obs,1} - t_{deliver,0} = \lambda_{SRC}$. In that case, $t_{deliver,1} = t_{obs,1}$. So:

$$Q \leq [\lambda_{SRC} + w_{RCV}]/\lambda_{SRC} \quad \text{(B.50)}$$

Since Q is an integer, we can take the floor on the right-hand side of the expression. Then:

$$Q|_{max} = \lfloor w_{RCV}/\lambda_{SRC} \rfloor + 1 \quad \text{(B.51)}$$

For a fast source clock:

$$\lambda_{SRC,fast} = \Lambda_{stream}/(1 + \rho_0) \quad \text{(B.52)}$$

For a slow receiver clock:

$$w_{RCV,slow} = (1 + \rho_0)W_{RCV} \quad \text{(B.53)}$$

Therefore, for the maximum value of Q:

$$Q|_{max} = \lfloor (W_{RCV}/\Lambda_{stream})(1 + \rho_0)^2 \rfloor + 1 \quad \text{(B.54)}$$

The minimum value of Q is derived as follows. Relation (B.47) can be re-expressed as:

$$t_{deliver,0} + (Q + 1)\lambda_{SRC} > t_{obs,1} + w_{RCV} \quad \text{(B.55)}$$

So:

$$Q > [(t_{obs,1} - t_{deliver,0}) + w_{RCV}]/\lambda_{SRC} - 1 \quad \text{(B.56)}$$

The right-hand side approaches its minimum value as $t_{obs,1} - t_{deliver,0}$ approaches 0. So:

$$Q > w_{RCV}/\lambda_{SRC} - 1 \quad \text{(B.57)}$$

Q is an integer strictly larger than $w_{RCV}/\lambda_{SRC} - 1$. The smallest integer that satisfies this relation is given by:

$$Q|_{min} = \lfloor w_{RCV}/\lambda_{SRC} \rfloor \quad \text{(B.58)}$$

For a slow source clock:

$$\lambda_{SRC,slow} = (1 + \rho_0)\Lambda_{stream} \quad \text{(B.59)}$$

For a fast receiver clock:

$$w_{RCV,fast} = W_{RCV}/(1 + \rho_0) \quad \text{(B.60)}$$

Therefore, for the minimum value of Q:

$$Q|_{min} = \lfloor (W_{RCV}/\Lambda_{stream})/(1 + \rho_0)^2 \rfloor \quad \text{(B.61)}$$

B.5.2. Expected local time of reception

The transmission times for the messages are related by the data introduction interval:

$$T_{SND,i} = T_{SND,0} + i\Lambda_{stream} \quad \text{(B.62)}$$

At the receiver, the relation among the messages is similar.

$$T_{RCV,E,i} = T_{RCV,E,0} + i\Lambda_{stream} \quad \text{(B.63)}$$

Using the analysis for single-message communication:

$$T_{RCV,E,i} = T_{SND,i} + R_{PP} \quad \text{(B.64)}$$

Let $T_{RCV,i}$ denote the actual time of reception for the i-th message. From the analysis of single-message communication, $T_{RCV,i}$ and $T_{RCV,E,i}$ are related as follows:

$$|T_{RCV,i} - T_{RCV,E,i}| \leq \Delta_{PP,RCV}|_{abs\text{-}max} \quad \text{(B.65)}$$

Re-expressing (B.65):

$$T_{RCV,E,i} - \Delta_{PP,RCV}|_{abs\text{-}max} \leq T_{RCV,i} \leq T_{RCV,E,i} + \Delta_{PP,RCV}|_{abs\text{-}max} \quad \text{(B.66)}$$

The stream as a whole should be received within the following local time interval:

$$[T_{RCV,E,0} - \Delta_{PP,RCV}|_{abs\text{-}max} , T_{RCV,E,K-1} + \Delta_{PP,RCV}|_{abs\text{-}max}] \quad \text{(B.67)}$$

Re-expressing (B.67):

$$[T_{RCV,E,0} - \Delta_{PP,RCV}|_{abs\text{-}max} , T_{RCV,E,0} + (K-1)\Lambda_{stream} + \Delta_{PP,RCV}|_{abs\text{-}max}] \quad \text{(B.68)}$$

B.5.3. Message reception rate

The Λ_{stream} communication parameter gives the nominal message reception rate for the stream in units of ticks per message. An important consideration for the processing of message streams is the relation between Λ_{stream} and $\Delta_{PP,RCV}|_{abs\text{-}max}$. As presented above, $\Delta_{PP,RCV}|_{abs\text{-}max}$ measures the uncertainty in the time of reception of each message. In particular, the total uncertainty in the time of reception for a particular message is $2\Delta_{PP,RCV}|_{abs\text{-}max}$ local clock ticks centered around the

expected time of reception. A message from a good source can be received at any of the $2\Delta_{PP,RCV}|_{abs\text{-}max} + 1$ triggering edges of the oscillator clock in the corresponding reception interval. Let Z denote the number of messages from a good source that can be received during a $2\Delta_{PP,RCV}|_{abs\text{-}max}$ interval. Then:

$$Z = \lfloor 2\Delta_{PP,RCV}|_{abs\text{-}max} / \Lambda_{stream} \rfloor + 1 \quad \text{(B.69)}$$

B.5.3.1. Non-overlapping reception intervals

If $\Lambda_{stream} > 2\Delta_{PP,RCV}|_{abs\text{-}max}$, the expected reception intervals for consecutive messages will not overlap or even coincide end-to-end (i.e., no shared triggering edges in consecutive expected reception intervals). For this case, Z = 1, which means that the messages of the stream are received as separate communications with no interaction.

B.5.3.2. Overlapping reception intervals

If $\Lambda_{stream} \leq 2\Delta_{PP,RCV}|_{abs\text{-}max}$, the expected reception intervals for consecutive messages will overlap or coincide at the ends. For this case, Z > 1, which means that the interaction between the messages must be taken into consideration. This is especially important for the diagnosis of timing errors.

B.5.4. Load size for a message reception buffer

We refer to the number of messages stored in a buffer as the **load** on the buffer. The function of the message receive buffer is to collect the messages received at the Computation Process. For single-message communication, it is expected that the processing of each message will begin at or before the next message is received. The same can occur for a message stream in which the reception intervals for consecutive messages do not overlap. In these cases, the load of the receive buffer is less than or equal to 1. From this point on, we only consider cases in which the processing of individual messages may begin after the reception of subsequent messages in the stream. This includes cases of overlapping and non-overlapping reception intervals.

Let $\Delta_{PROC,begin}$ denote the delay in the beginning of processing of a message with respect to the corresponding expected time of reception. We assume that the interval between the beginning of processing of consecutive messages is the same as the data introduction interval for the message stream, Λ_{stream}. $T_{PROC,i}$ denotes the local time at the beginning of processing for message i.

$$T_{PROC,i} = T_{RCV,E,i} + \Delta_{PROC,begin} \quad \text{(B.70)}$$

B.5.4.1. Combined message synchronization and buffering

Figure B.3 illustrates the interconnection of functions for this case. CLK_{Rx} denotes the oscillator clock at the receiving node. STB_{Rx} denotes the strobe signal indicating that a new message is ready. The Link Receiver transfers the messages to the Receive Buffer as soon as they are ready. The output of the receiver is assumed to be asynchronous with respect to the oscillator clock. The Receive Buffer is an asynchronous FIFO, which means that the push (i.e.,

write) and pop (i.e., remove) action signals may be synchronous with respect to different clock signals. In effect, in addition to being a buffer, the asynchronous FIFO serves as a signal synchronizer for data crossing from one clock domain to the other. Note that the data is read for computation one tick before it is popped from the receive buffer.

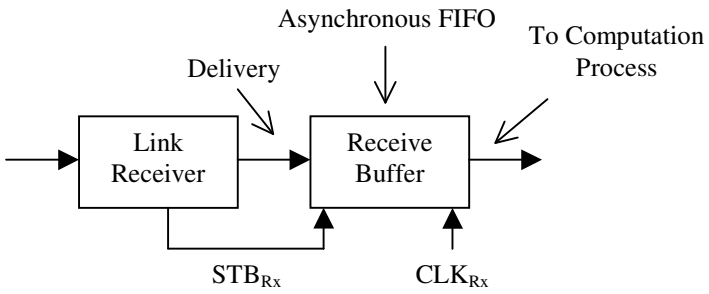


Figure B.3: Reception using combined message synchronization and buffering

In order to ensure a read-after-write sequence at the Receive Buffer during normal operation, the reading of a particular message by the Computation Process should be triggered after the end of the corresponding reception interval. The following relation must hold in order to satisfy this property.

$$\Delta_{PROC,begin} > \Delta_{PP,RCV}|_{abs\text{-}max} \tag{B.71}$$

Again, note that the pop takes place one tick after the start of processing for each message. $t_{deliver,i}$ denotes the real time at which message i is written to the buffer. Note that a message gets pushed at the same time that it is delivered. With λ_{SRC} denoting the data introduction interval at the source node, $t_{deliver,i}$ is given by the following equation.

$$t_{deliver,i} = t_{deliver,0} + i\lambda_{SRC} \tag{B.72}$$

Let $t_{pop,i}$ denote the real time at which message i is popped from the buffer. The pop times for the Computation Process are given by the following relation, with λ_{RCV} equal to the data introduction interval at the Computation Process.

$$t_{pop,i} = t_{pop,0} + i\lambda_{RCV} \tag{B.73}$$

Let $Q_{deliver}(t)$ denote the number of delivered messages by time t. $Q_{pop}(t)$ denotes the number of popped messages by time t. $Q_{Async\text{-}Buffer}(t)$ denotes the number of messages held by the asynchronous Receive Buffer at time t.

For $Q_{deliver}(t)$:

$$Q_{deliver}(t) = \begin{cases} 0, & \text{for } t < t_{deliver,0} \\ \lfloor [(t - t_{deliver,0})/\lambda_{SRC}] + 1 \rfloor, & \text{for } t_{deliver,0} \le t \le t_{deliver,0} + (K-1)\lambda_{SRC} \\ K, & \text{for } t > t_{deliver,0} + (K-1)\lambda_{SRC} \end{cases} \tag{B.74}$$

For $Q_{pop}(t)$:

$$Q_{pop}(t) = \begin{cases} 0, & \text{for } t < t_{pop,0} \\ \lfloor [(t - t_{pop,0})/\lambda_{RCV}] + 1 \rfloor, & \text{for } t_{pop,0} \le t \le t_{pop,0} + (K\text{-}1)\lambda_{RCV} \\ K, & \text{for } t > t_{pop,0} + (K\text{-}1)\lambda_{RCV} \end{cases} \quad \text{(B.75)}$$

For $Q_{Asyn\text{-}Buffer}(t)$:

$$Q_{Asyn\text{-}Buffer}(t) = Q_{deliver}(t) - Q_{pop}(t) \quad \text{(B.76)}$$

To determine the maximum load for the receive buffer, we consider the case of a fast source clock and a slow receiver clock. Thus:

$$\lambda_{SRC} = \lambda_{SRC,fast} = \Lambda_{stream}/(1 + \rho_0) \quad \text{(B.77)}$$

$$\lambda_{RCV} = \lambda_{RCV,slow} = (1 + \rho_0)\Lambda_{stream} \quad \text{(B.78)}$$

Assume that the first message is delivered at the earliest possible time. That is:

$$t_{deliver,0} = c_{RCV}(T_{RCV,E,0} - \Delta_{PP,RCV}|_{abs\text{-}max}) \quad \text{(B.79)}$$

The time of the first pop action is:

$$\begin{aligned} t_{pop,0} &= c_{RCV}(T_{RCV,E,0} + \Delta_{PROC,begin} + 1) \\ &= t_{deliver,0} + (1 + \rho_0)(\Delta_{PP,RCV}|_{abs\text{-}max} + \Delta_{PROC,begin} + 1) \end{aligned} \quad \text{(B.80)}$$

Since the source has a faster clock, the number of buffered messages can increase up to the instant the last message is delivered (i.e., $t = t_{deliver,K\text{-}1} = t_{deliver,0} + (K\text{-}1)\lambda_{SRC}$). Thus, the maximum buffer load is given by $Q_{Async\text{-}Buffer}$ evaluated at $t_{deliver,K\text{-}1}$.

$$\begin{aligned} Q_{Asyn\text{-}Buffer}(t)|_{max} &= Q_{Asyn\text{-}Buffer}(t_{deliver,K\text{-}1}) \\ &= K - \lfloor [(K\text{-}1)\lambda_{SRC,fast} - (1 + \rho_0)(\Delta_{PP,RCV}|_{abs\text{-}max} + \Delta_{PROC,begin} + 1)]/\lambda_{RCV,slow} + 1 \rfloor \\ &= K - \lfloor (K\text{-}1)/(1 + \rho_0)^2 - (\Delta_{PP,RCV}|_{abs\text{-}max} + \Delta_{PROC,begin} + 1)/\Lambda_{stream} + 1 \rfloor \end{aligned} \quad \text{(B.81)}$$

B.5.4.2. Separate message synchronization and buffering

Figure B.4 illustrates the interconnection of functions for this case. The receiver is assumed to hold the message until it is processed by the synchronizer. For this synchronization mechanism, the input rate must be slower than the local clock frequency to ensure at least one triggering edge of the oscillator clock per delivered message. Thus, Λ_{stream} must be larger than 1. Since Λ_{stream} is an integer, its value must at least 2 (i.e., $\Lambda_{stream} \ge 2$).

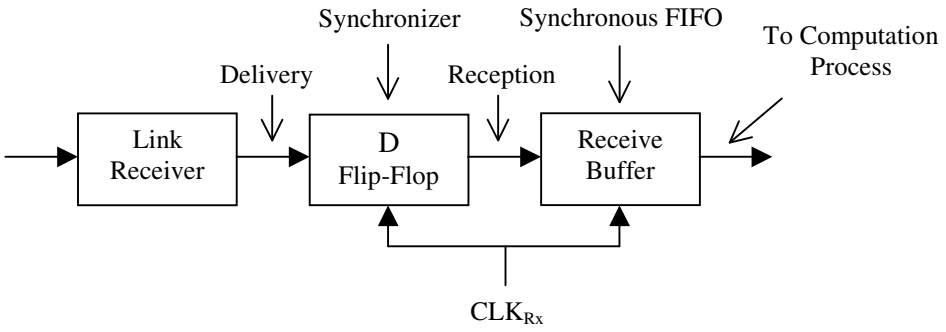


Figure B.4: Reception using separate message synchronization and buffering

This configuration differs from the one using the asynchronous FIFO in that the synchronization is performed by a dedicated synchronizer. At a minimum, this element introduces a one-tick delay in the transfer of messages from the Link Receiver to the Receive Buffer. The worst-case delay in storing the message in the buffer is two oscillator clock ticks. Therefore, compared to the timing of the circuit with the asynchronous FIFO, the writing of streamed messages to the synchronous FIFO buffer begins at least 1 local tick later and can end up to 2 oscillator clock ticks later.

To determine the maximum load for the receive buffer, we consider the case of a fast source clock and a slow receiver clock. The maximum load is assessed at the earliest time at which the last message of the input stream can be written into the buffer. Let $t_{write,0}$ denote the earliest real time at which a received message is written to the buffer.

$$t_{write,0} = c_{RCV}(T_{RCV,E,0} - \Delta_{PP,RCV}|_{abs\text{-}max} + 1) = t_{deliver,0} + (1 + \rho_0) \quad \text{(B.82)}$$

The delivery time for the last message is:

$$t_{deliver,K-1} = t_{deliver,0} + (K-1)\lambda_{SRC,fast} \quad \text{(B.83)}$$

After delivery, the message must be synchronized and written to the buffer. In the fastest case, the delivered message is immediately read by the synchronizer and presented to the buffer for loading, which will then occur 1 tick later.

$$t_{write,K-1} = t_{deliver,K-1} + (1 + \rho_0) = t_{deliver,0} + (K-1)\lambda_{SRC,fast} + (1 + \rho_0) \quad \text{(B.84)}$$

Let $Q_{Sync\text{-}Buffer}(t)$ denote the number of messages held by the synchronous receive buffer at time t. The maximum load is given by:

$$\begin{aligned} Q_{Sync\text{-}Buffer}(t)|_{max} &= Q_{Sync\text{-}Buffer}(t_{write,K-1}) \\ &= K - Q_{pop}(t_{write,K-1}) \\ &= K - \lfloor [(t_{write,K-1} - t_{pop,0})/\lambda_{RCV,slow}] + 1 \rfloor \\ &= K - \lfloor (K-1)/(1 + \rho_0)^2 - (\Delta_{PP,RCV}|_{abs\text{-}max} + \Delta_{PROC,begin})/\Lambda_{stream} + 1 \rfloor \end{aligned} \quad \text{(B.85)}$$

Appendix C. Analysis of the clock synchronization protocols

This appendix examines the timing aspects for the local-time synchronization scheme. The diagnostic system works in close coordination with the clock synchronization system to determine the status of the bus and to specify the nodes eligible to participate in clock synchronization operations. That aspect of the ROBUS is outside the scope of this appendix. The analysis presented here uses the fundamental fault-tolerance concepts presented in Appendix A and the point-to-point communication concepts presented in Appendix B.

C.1. Clock synchronization system

Each ROBUS node is driven by an independent, free-running physical oscillator and a local-time clock. The oscillators are characterized by a nominal frequency (denoted by f_0) and a bounded drift rate with respect to real time (denoted by ρ_0). ρ_0 is a small, positive, real-valued, unitless constant. τ_0 denotes the nominal tick duration measured in seconds: 1 nominal tick = τ_0 seconds = $1/f_0$ seconds. Let τ_x denote the actual tick duration for oscillator x. The bound on the drift rate of x can be expressed as follows:

$$\tau_0/(1 + \rho_0) \leq \tau_x \leq (1 + \rho_0)\tau_0 \qquad \text{(C.1)}$$

So, an actual oscillator has a tick duration between $1/(1 + \rho_0)$ and $(1 + \rho_0)$ nominal ticks.

The local-time clock of a node is essentially a counter driven by the local physical oscillator. The local time is equal to the state of the counter. Resetting the counter sets the local time to 0.

The clock synchronization system enables the nodes to use the local time as a reference for the coordination of distributed operations. A basic requirement for proper distributed coordination is that the relative clock skews remain within known bounds. The **relative skew** between two clocks is the real time elapsed from the instant one clock makes a particular state transition (i.e., the count reaches a particular value) until the other clock makes the same transition. In general, the relative skew between two events is equal to the real time elapsed between the occurrence of the events. Bounded relative skew is achieved by the generation and preservation of approximate real-time agreement on the transitions of the local-time clock. The synchronization protocols deliver high-precision distributed events used as references to reset the local-time clocks. The state of a local-time clock indicates the time elapsed since the last synchronization-reset event. The bound on the relative skew between synchronized clocks is tightest at the time of the synchronization reset. After the reset, the local times can drift apart from each other and from real time at rates determined by the drift rates of the oscillators. The clocks are synchronized at regular time intervals in order to ensure that the relative skews remain within known bounds.

Figure C.1 illustrates the conceptual mode transitions for the clock synchronization system. Normally there is a clique executing the Synchronization Preservation (SP) protocol to ensure that their relative local-time skews remain within known bounds. Nodes in this mode are said to be in a **synchronized state**. A goal of every node is to reach and remain in this state. In the context of the synchronization system, nodes operating in a mode other than Synchronization Preservation are referred to as **recovering** nodes. After a power-on enable or the detection of a failure, a node examines the activity on the bus. If a clique is found, the recovering node executes the Synchronization Acquisition (SA) mode in order to synchronize its local time to the

time of the clique. If a clique is not found, the recovering node transitions to the Initial Synchronization (IS) mode. After achieving synchronization, the recovering node executes the Synchronization Preservation protocol.

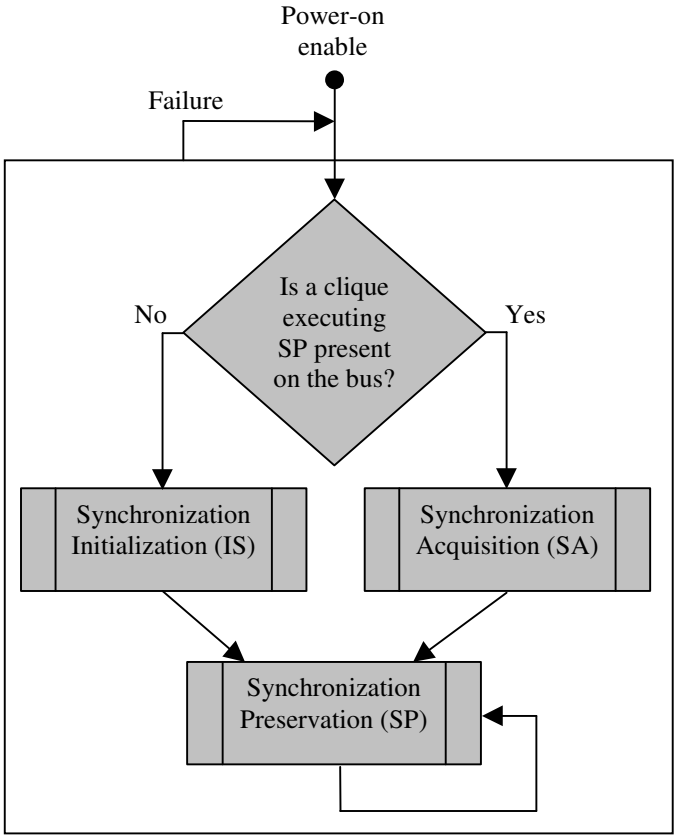


Figure C.1: Conceptual mode transitions for the clock synchronization system

At the time of entry into the Synchronization Acquisition mode, the recovering node is in an **asynchronous state** in which there is no significant relation between its local time and the local time of the clique. The recovering node uses an Accept function to capture the synchronization events in the agreement propagation phase of the Synchronization Preservation protocol. This requires that the Accept function only receive synchronization messages from the same execution of the protocol. This is accomplished by enabling the Accept function after a frame synchronization step in which the gap between executions of the Synchronization Preservation protocol is found.

In general, a group of nodes enters the Initial Synchronization mode within a time interval of known bounded duration. When a recovering node enters this mode, it expects that there is at least one node of the opposite kind that also makes the transition within the bounded time interval. This interval duration is in effect a bound on the relative local-time skew for the initializing nodes. Before the execution of the synchronization protocol, these nodes are said to be in an **unsynchronized state** since the initial skew bound can be relatively large compared to the skew after the execution of the protocol.

Figure C.2 illustrates how the mode transitions are related in time. A group of nodes enters Initial Synchronization with a large bound on the relative skew, denoted by π_{IS}. At the end of the protocol execution, the local time is set to 0 with the bound on the relative skew reduced to the level required for normal operation, denoted by π_{SP}. At local time T_{SP}, the Synchronization Preservation protocol is executed to ensure that the skew remains within the expected bound. This cyclic operation continues until a failure occurs or the system is shut down. A recovering node in Synchronization Acquisition trying to synchronize to the clique executes the Frame Synchronization (FS) protocol followed by the Synchronization Capture (SC) protocol. The duration of the Frame Synchronization protocol execution depends on factors like the total number of nodes of the opposite kind, the number of untrustworthy nodes of the opposite kind active on the bus, the bound on the relative local-time skew of the nodes, and the position of the start of the protocol relative to local time of the clique nodes. Synchronization Capture is enabled immediately after the execution of Frame Synchronization is complete. The relative skew achieved by Synchronization Acquisition is within the bounds of the skew for normal operation.

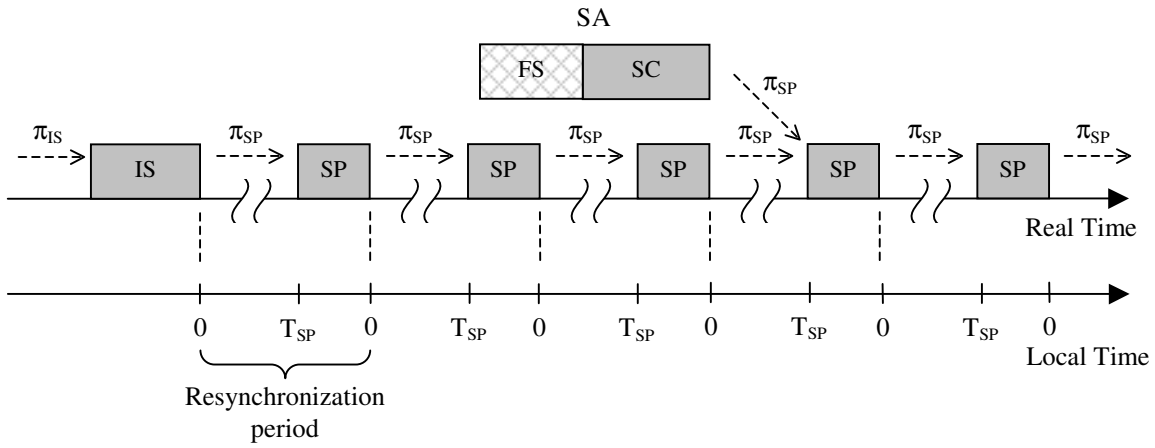


Figure C.2: Timing of mode transitions for the clock synchronization system

The Initial Synchronization, Synchronization Preservation, and Synchronization Capture protocols are based on the same theory of distributed computation using Accept functions to process timing events. Figure C.3 illustrates the message flow graph examined in this appendix. This graph includes all the processes and messages required for the three protocols.

C.2. Timing model

This section describes how the ROBUS nodes are modeled for the analysis of the synchronization protocols.

C.2.1. Computation Module

The Computation Module is modeled in terms of two components: Computation Process and Send Process. The **Computation Process** handles the processing of received messages according to the requirements of the protocol being executed. The **Send Process** handles the timing and formatting requirements for the output messages.

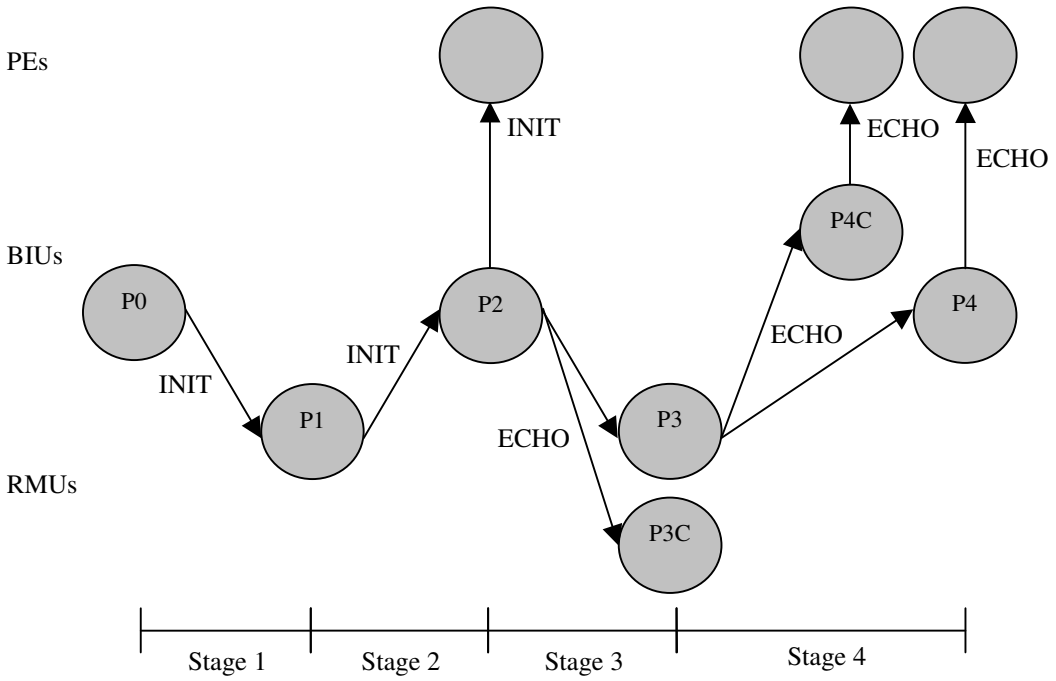


Figure C.3: Combined message flow graph for the analysis of the synchronization protocols

The Computation Process has two sub-processes: Receive and Accept. The **Receive Process** provides timing delay and in-line error detection for the received messages. The input-output delay of the Received Process depends on the synchronization process being executed. For a particular process, every input has the same input-output delay. This behavior preserves the relative positions of the received synchronization messages, which allows the Accept function to properly read the relative skews of the received timing events. The delay of the Receive Process depends mainly on the uncertainty in the time of reception and on the time required to process the messages for error detection and diagnosis. The **Accept Process** performs the voting of the set of received messages to produce a single result event. The delay of the Accept Process depends on the protocol being executed. The Accept Process produces an output event with a predetermined delay with respect to the time at which it receives the input event to be selected. For the events not selected, the Accept function appears to have a variable input-output delay.

For each of the synchronization processes, the Computation Process has a fixed delay from the time when the event to be selected is received to the time when the Accept output is asserted. The delays of the Receive Process and the Accept Process are combined into a single parameter denoted by A, which is measured in units of local clock ticks.

The Send Process handles the transmission of messages. The process delay depends on the protocol and process being executed. B denotes the send delay with respect to the process-triggering event, and it is measured in units of local clock ticks.

C.2.2. Communication Module

The behavior of the Communication Module is independent of the protocol being executed by

the Computation Module. Appendix B presents the timing model for point-to-point communication.

C.3. First stage

Figure C.4 illustrates the detailed message flow graph for stage 1 in a 3x3 system (i.e., 3 BIUs and 3 RMUs).

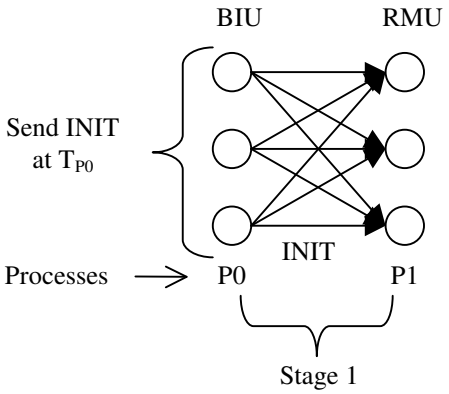


Figure C.4: Detailed message flow graph for stage 1 in a 3x3 system

C.3.1. Expected time of reception for process P1

The analysis for point-to-point communication presented in the Appendix B can be leveraged for the problem of determining the local time range of reception of INIT messages in process P1. This is covered in section A.10.2.1 of this appendix for the Initial Synchronization and Synchronization Preservation protocols.

C.3.2. Bound on the observed relative skew of received messages for process P1

Let $\Pi_{P1,RCV}$ denote the bound on the relative skew observed in process P1 for the received messages from process P0 at trustworthy BIUs. $\Pi_{P1,RCV}$ is measured in local clock ticks. $\Pi_{P1,RCV}$ is used to check for agreement among the received inputs and also to check agreement with the result of the Accept output.

Let T_{P0} denotes the local time at which a BIU node sends the INIT message in process P0 (i.e., the local time when the source's Computation Module signals the Communication Module to send the INIT message). $t_{P0,l}$ and $t_{P0,h}$ denote the earliest and latest real times, respectively, at which the trustworthy BIU nodes send INIT in process P0. Let π_{P0} denote the bound on the relative local-time skew for the trustworthy BIUs. π_{P0} is assumed to apply for the duration of the protocol execution. π_{P0} also bounds the precision with which the trustworthy BIU nodes send the INIT messages.

$$\pi_{P0} = t_{P0,h} - t_{P0,l} \qquad \text{(C.2)}$$

Let $t_{P1,RCV,l}$ and $t_{P1,RCV,h}$ denote the earliest and latest real times, respectively, at which an INIT message from a trustworthy BIU node can be received in process P1 at the trustworthy RMUs.

$$t_{P1,RCV,l} = t_{P0,l} + r_{PP,l} \tag{C.3}$$

$$t_{P1,RCV,h} = t_{P0,h} + r_{PP,h} \tag{C.4}$$

Let $T_{P1,RCV,l}$ and $T_{P1,RCV,h}$ denote the earliest and latest local times, respectively, at which a node in process P1 can receive messages from process P0 at trustworthy BIU nodes. $\Delta_{P1,RCV}$ denotes the measured skew between the earliest and latest received messages from trustworthy BIU nodes (i.e., $\Delta_{P1,RCV} = T_{P1,RCV,h} - T_{P1,RCV,l}$). We need to determine the maximum value of $\Delta_{P1,RCV}$. Using (C.3), (C.4), and the local-clock function (introduced in Appendix B):

$$c_{RCV}(T_{P1,RCV,h}) - c_{RCV}(T_{P1,RCV,l}) \leq t_{P1,RCV,h} - t_{P1,RCV,l} \tag{C.5}$$

From the constraint that the drift rate of the local clocks be ρ_0-bounded and the definition of $\Delta_{P1,RCV}$:

$$\Delta_{P1,RCV}/(1 + \rho_0) \leq c_{RCV}(T_{P1,RCV,h}) - c_{RCV}(T_{P1,RCV,l}) \tag{C.6}$$

Combining (C.5) and (C.6), and using the fact that $\Delta_{P1,RCV}$ is an integer:

$$\Delta_{P1,RCV} \leq \lfloor (1 + \rho_0)(t_{P1,RCV,h} - t_{P1,RCV,l}) \rfloor \tag{C.7}$$

$\Pi_{P1,RCV}$ is given by the maximum value of $\Delta_{P1,RCV}$:

$$\Pi_{P1,RCV} = \Delta_{P1,RCV}|_{max} = \lfloor (1 + \rho_0)(t_{P1,RCV,h} - t_{P1,RCV,l}) \rfloor = \lfloor (1 + \rho_0)(\pi_{P0} + e_{PP}) \rfloor \tag{C.8}$$

C.3.3. Relative skew of the Accept outputs for process P1

Let A_{P1} denote the delay (in local-clock ticks) of the Computation Process in process P1 measured from the local time of reception of the selected message until the Accept output is asserted. $t_{P1,A,l}$ and $t_{P1,A,h}$ denote the earliest and latest real times, respectively, at which an Accept output in process P1 at the trustworthy RMUs can be asserted.

$$t_{P1,A,l} = t_{P0,l} + r_{PP,l} + A_{P1}/(1 + \rho_0) \tag{C.9}$$

$$t_{P1,A,h} = t_{P0,h} + r_{PP,h} + (1 + \rho_0)A_{P1} \tag{C.10}$$

Therefore, the Accept functions of the trustworthy RMU nodes assert their outputs during a real-time interval with the following duration:

$$\begin{aligned} t_{P1,A,h} - t_{P1,A,l} &= [\pi_{P0} + r_{PP,h} + (1 + \rho_0)A_{P1}] - [r_{PP,l} + A_{P1}/(1 + \rho_0)] \\ &= \pi_{P0} + e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)]A_{P1} \end{aligned} \tag{C.11}$$

Let AEV_P1 denote the set of asymmetric BIU eligible voters in process P1 at a trustworthy RMU node. |AEV_P1| denotes the cardinality of AEV_P1. $\pi_{P1,A}$ denotes the bound on the real-

time relative skew of the Accept outputs in process P1 at the trustworthy RMUs. If |AEV_P1| = 0 for each trustworthy RMU node, they essentially accept on the same message.

$$\pi_{P1,A}|_{|AEV\_P1|=0} = e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)]A_{P1} \quad \text{(C.12)}$$

If |AEV_P1| ≠ 0 for some trustworthy RMU, all we know with certainty is that the RMU nodes accept on a message from a trustworthy BIU node or a message from an untrustworthy BIU node flanked by messages from trustworthy BIU nodes.

$$\pi_{P1,A}|_{|AEV\_P1|\neq 0} = \pi_{P0} + e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)]A_{P1} \quad \text{(C.13)}$$

From this point on, unless otherwise stated:

$$\pi_{P1,A} = \pi_{P1,A}|_{max} = \pi_{P1,A}|_{|AEV\_P1|\neq 0} \quad \text{(C.14)}$$

C.4. Second stage

Figure C.5 illustrates the detailed message flow graph for stage 1 and 2 in a 3x3 system.

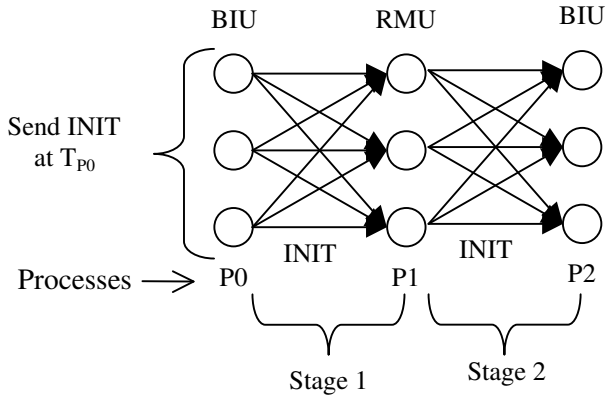


Figure C.5: Detailed message flow graph for stages 1 and 2 in a 3x3 system

C.4.1. Effective reception delay for process P2

Let B_{P0} denote the send delay for process P0. We want to compute the effective reception delay for process P2. In general, this delay is measured from the time of some local event to the time of reception. We use T_{P0}, the local time of transmission of the INIT message in process P0, as the local reference event to measure the reception delay from process P0 to process P2. Note that instead of the start of the protocol, we choose the send time for process P0 as the reference time to measure the reception delay. This approach enables the analysis of the synchronization protocols independently of B_{P0}. B_{P0} is computed based on a single-stage point-to-point synchronous communication model (see section A.10.2.1 of this appendix).

We need to determine the earliest and latest real times of reception for process P2. Let B_{P1} denote the send delay for process P1. $t_{P2,RCV,l}$ and $t_{P2,RCV,h}$ denote the earliest and latest real times,

respectively, at which an INIT message from a trustworthy RMU node can be received in process P2 at the trustworthy BIUs.

$$t_{P2,RCV,l} = t_{P1,A,l} + B_{P1}/(1 + \rho_0) + r_{PP,l} = t_{P0,l} + 2r_{PP,l} + (A_{P1} + B_{P1})/(1 + \rho_0) \quad \text{(C.15)}$$

$$t_{P2,RCV,h} = t_{P1,A,h} + (1 + \rho_0)B_{P1} + r_{PP,h} = t_{P0,l} + \pi_{P0} + 2r_{PP,h} + (1 + \rho_0)(A_{P1} + B_{P1}) \quad \text{(C.16)}$$

$r_{P0\text{-}P2,l}$ denotes the minimum effective message-reception delay for INIT messages in process P2 and is measured from the latest time at which the trustworthy BIU nodes can send INIT to the earliest time at which the BIU nodes can receive INIT messages from the trustworthy RMU nodes.

$$r_{P0\text{-}P2,l} = t_{P2,RCV,l} - t_{P0,h} = 2r_{PP,l} + (A_{P1} + B_{P1})/(1 + \rho_0) - \pi_{P0} \quad \text{(C.17)}$$

$r_{P0\text{-}P2,h}$ denotes the maximum effective message-reception delay for INIT messages in process P2 and is measured from the earliest time at which the trustworthy BIU nodes can send INIT to the latest time at which the BIU nodes can receive INIT messages from the trustworthy RMU nodes.

$$r_{P0\text{-}P2,h} = t_{P2,RCV,h} - t_{P0,l} = \pi_{P0} + 2r_{PP,h} + (1 + \rho_0)(A_{P1} + B_{P1}) \quad \text{(C.18)}$$

The expected reception delay for process P2 is:

$$R_{P0\text{-}P2} = IMP(r_{P0\text{-}P2,l}\ ,\ r_{P0\text{-}P2,h}) \quad \text{(C.19)}$$

The IMP function is defined in Appendix B.

The total effective uncertainty in the real time of reception of the INIT messages in process P2 is:

$$r_{P0\text{-}P2,h} - r_{P0\text{-}P2,l} = 2\pi_{P0} + 2e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](A_{P1} + B_{P1}) \quad \text{(C.20)}$$

C.4.2. Expected time of reception for process P2

The BIU nodes expect to receive INIT messages at local time $T_{P2,RCV,E}$.

$$T_{P2,RCV,E} = T_{P0} + R_{P0\text{-}P2} \quad \text{(C.21)}$$

The real-time error for $T_{P2,RCV,E}$ is bounded as follows. A BIU node will receive an INIT message from a trustworthy RMU node no earlier than $\mu_{P0\text{-}P2,l}$ nominal ticks from $T_{P2,RCV,E}$.

$$\mu_{P0\text{-}P2,l} = (1 + \rho_0)R_{P0\text{-}P2} - r_{P0\text{-}P2,l} \quad \text{(C.22)}$$

A BIU node will receive an INIT message from a trustworthy RMU node no later than $\mu_{P0\text{-}P2,h}$ nominal ticks from $T_{P2,RCV,E}$.

$$\mu_{P0\text{-}P2,h} = r_{P0\text{-}P2,h} - R_{P0\text{-}P2}/(1 + \rho_0) \quad \text{(C.23)}$$

Let $T_{P2,RCV}$ denote the actual local time at a BIU node when an INIT message from a trustworthy RMU node is received. In addition, let $\Delta_{P2,RCV}$ denote the local-time error in $T_{P2,RCV}$.

$$\Delta_{P2,RCV} = T_{P2,RCV} - T_{P2,RCV,E} \quad \text{(C.24)}$$

We want to determine a bound for the local-time error in the actual time of reception in process P2, denoted by $\Delta_{P2,RCV}|_{max}$.

$$|T_{P2,RCV} - T_{P2,RCV,E}| \leq \Delta_{P2,RCV}|_{max} \quad \text{(C.25)}$$

$\Delta_{P2,RCV}|_{max}$ is derived as follows. We know that the difference between the expected and the actual time of reception at a BIU node for INIT messages from the trustworthy RMU nodes is bounded by $\mu_{P0\text{-}P2,l}$ and $\mu_{P0\text{-}P2,h}$, such that:

$$c_{RCV}(T_{P2,RCV,E}) - \mu_{P0\text{-}P2,l} \leq c_{RCV}(T_{P2,RCV}) \leq c_{RCV}(T_{P2,RCV,E}) + \mu_{P0\text{-}P2,h} \quad \text{(C.26)}$$

So:

$$|c_{RCV}(T_{P2,RCV}) - c_{RCV}(T_{P2,RCV,E})| \leq \max(\mu_{P0\text{-}P2,l}\ ,\ \mu_{P0\text{-}P2,h}) \quad \text{(C.27)}$$

From the constraint that the local clocks be ρ-bounded and (C.24):

$$|\Delta_{P2,RCV}|/(1 + \rho_0) \leq |c_{RCV}(T_{P2,RCV}) - c_{RCV}(T_{P2,RCV,E})| \quad \text{(C.28)}$$

Combining (C.27) and (C.28):

$$|\Delta_{P2,RCV}| \leq (1 + \rho_0)\max(\mu_{P0\text{-}P2,l}\ ,\ \mu_{P0\text{-}P2,h}) \quad \text{(C.29)}$$

Since $\Delta_{P2,RCV}$ is an integer:

$$|\Delta_{P2,RCV}| \leq \lfloor (1 + \rho_0)\max(\mu_{P0\text{-}P2,l}\ ,\ \mu_{P0\text{-}P2,h}) \rfloor \quad \text{(C.30)}$$

Therefore:

$$\Delta_{P2,RCV}|_{max} = \lfloor (1 + \rho_0)\ \max(\mu_{P0\text{-}P2,l}\ ,\ \mu_{P0\text{-}P2,h}) \rfloor \quad \text{(C.31)}$$

C.4.3. Bound on the observed relative skew of received messages for process P2

Let $\Pi_{P2,RCV}$ denote the bound on the relative skew observed in process P2 for the received messages from process P1 at trustworthy RMUs. $\Pi_{P2,RCV}$ is measured in local clock ticks. $\Pi_{P2,RCV}$ is used to check for agreement among the received inputs and also to check agreement with the result of the Accept output.

The worst case relative skew for received messages occurs when there are asymmetric eligible voters in process P1 at the trustworthy RMU nodes. The bound on the relative skew of the Accept outputs in process P1 is $\pi_{P1,A}$. The additional uncertainty in the reception delay measured from the time of the Accept outputs to the time of reception in process P2 is $e_{PP} + [(1+\rho_0) - 1/(1+\rho_0)]B_{P1}$.

$$\Pi_{P2,RCV} = \lfloor (1 + \rho_0)(t_{P2,RCV,h} - t_{P2,RCV,l}) \rfloor$$

$$= \lfloor (1 + \rho_0)\{\pi_{P1,A} + e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)]B_{P1}\} \rfloor$$

$$\Pi_{P2,RCV} = \lfloor (1 + \rho_0)\{\pi_{P0} + 2e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](A_{P1} + B_{P1})\} \rfloor \quad \text{(C.32)}$$

C.4.4. Relative skew of the Accept outputs for process P2

Let AEV_P2 denote the set of asymmetric RMU eligible voters in process P2 at a trustworthy BIU node. Let $\pi_{P2,A}$ denote the bound on the real-time relative skew of the Accept outputs in process P2 at trustworthy BIUs. A_{P2} denotes the delay (in local-clock ticks) of the Computation Process in process P2 measured from the local time of reception of the selected message to the local time when the Accept output is asserted. If |AEV_P1| = 0 for each trustworthy RMU node, the trustworthy BIU nodes may have asymmetric RMU nodes in their sets of eligible voters for process P2 (i.e., |AEV_P2| ≠ 0 for some trustworthy BIUs). In this case, the trustworthy BIU nodes accept within the time range delimited by messages from trustworthy RMU nodes.

$$\pi_{P2,A}|_{|AEV\_P2| \neq 0} = \pi_{P1,A}|_{|AEV\_P1| = 0} + e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](B_{P1} + A_{P2})$$

$$= 2e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](A_{P1} + B_{P1} + A_{P2}) \quad \text{(C.33)}$$

If |AEV_P1| ≠ 0 for some trustworthy RMU nodes, the trustworthy BIU nodes do not have asymmetric RMU nodes in their sets of eligible voters for process P2 (i.e., |AEV_P2| = 0 at each trustworthy BIU). In this case, the BIU nodes essentially accept on the same message.

$$\pi_{P2,A}|_{|AEV\_P2| = 0} = e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)]A_{P2} \quad \text{(C.34)}$$

From this point on, unless otherwise stated:

$$\pi_{P2,A} = \pi_{P2,A}|_{max} = \pi_{P2,A}|_{|AEV\_P2| \neq 0} \quad \text{(C.35)}$$

C.5. Third stage

Figure C.6 illustrates the detailed message flow graph up to stage 3 for a 3x3 system.

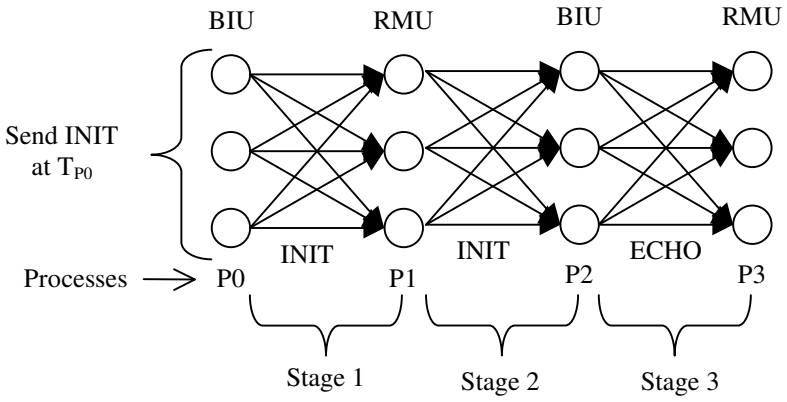


Figure C.6: Detailed message flow graph for stages 1 through 3 in a 3x3 system

C.5.1. Effective reception delay for process P3

We need to determine the earliest and latest real times of reception of ECHO messages in process P3. Let $T_{P1,A,i}$ denote the local time at RMU node i when it asserts the output of its Accept function in process P1. Let $t_{P1,A,l}$ and $t_{P1,A,h}$ denote the earliest and latest real times, respectively, at which the trustworthy RMUs can assert the Accept outputs in process P1.

$$\pi_{P1,A} = t_{P1,A,h} - t_{P1,A,l} \tag{C.36}$$

Let B_{P2} denote the send delay for process P2. $t_{P3,RCV,l}$ and $t_{P3,RCV,h}$ denote the earliest and latest real times, respectively, at which ECHO messages from trustworthy BIU nodes can be received by an RMU node in process P3.

$$t_{P3,RCV,l} = t_{P1,A,l} + 2r_{PP,l} + (B_{P1} + A_{P2} + B_{P2})/(1 + \rho_0) \tag{C.37}$$

$$t_{P3,RCV,h} = t_{P1,A,h} + 2r_{PP,h} + (1 + \rho_0)(B_{P1} + A_{P2} + B_{P2}) \tag{C.38}$$

$r_{P1\text{-}P3,l}$ denotes the minimum effective message-reception delay for ECHO messages in process P3 and is measured from the latest time at which trustworthy RMU nodes can assert their Accept(INIT) output to the earliest time at which the RMU nodes receive ECHO messages from the trustworthy BIU nodes.

$$r_{P1\text{-}P3,l} = t_{P3,RCV,l} - t_{P1,A,h} = 2r_{PP,l} + (B_{P1} + A_{P2} + B_{P2})/(1 + \rho_0) - \pi_{P1,A} \tag{C.39}$$

$r_{P1\text{-}P3,h}$ denotes the maximum effective message-reception delay for ECHO messages in process P3 and is measured from the earliest time at which the trustworthy RMU nodes can assert its Accept(INIT) output to the latest time at which the RMU nodes can receive ECHO messages from the trustworthy BIU nodes.

$$r_{P1\text{-}P3,h} = t_{P3,RCV,h} - t_{P1,A,l} = \pi_{P1,A} + 2r_{PP,h} + (1 + \rho_0)(B_{P1} + A_{P2} + B_{P2}) \tag{C.40}$$

The expected reception delay for process P3 is:

$$R_{P1\text{-}P3} = IMP(r_{P1\text{-}P3,l}\ ,\ r_{P1\text{-}P3,h}) \tag{C.41}$$

The effective uncertainty in the real time of reception of the ECHO messages in process P3 is:

$$r_{P1\text{-}P3,h} - r_{P1\text{-}P3,l} = 2\pi_{P1,A} + 2e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](B_{P1} + A_{P2} + B_{P2}) \tag{C.42}$$

C.5.2. Expected time of reception for process P3

RMU node i expects to receive ECHO messages at local time $T_{P3,RCV,E,i}$.

$$T_{P3,RCV,E,i} = T_{P1,A,i} + R_{P1\text{-}P3} \tag{C.43}$$

The real-time error for $T_{P3,RCV,E,i}$ is bounded as follows. RMU node i will receive an ECHO message from a trustworthy BIU node no earlier than $\mu_{P1\text{-}P3,l}$ nominal ticks from $T_{P3,RCV,E,i}$.

$$\mu_{P1\text{-}P3,l} = (1 + \rho_0)R_{P1\text{-}P3} - r_{P1\text{-}P3,l} \quad \text{(C.44)}$$

RMU node i will receive an ECHO message from a trustworthy BIU node no later than $\mu_{P1\text{-}P3,h}$ nominal ticks from $T_{P3,RCV,E,i}$.

$$\mu_{P1\text{-}P3,h} = r_{P1\text{-}P3,h} - R_{P1\text{-}P3}/(1 + \rho_0) \quad \text{(C.45)}$$

We want to determine the maximum local-time error for the actual time of reception at the RMU nodes, denoted by $\Delta_{P3,RCV}|_{max}$.

$$|T_{P3,RCV} - T_{P3,RCV,E}| \leq \Delta_{P3,RCV}|_{max} \quad \text{(C.46)}$$

Following the analysis for process P2:

$$\Delta_{P3,RCV}|_{max} = \lfloor (1 + \rho_0)\max(\mu_{P1\text{-}P3,l}\ ,\ \mu_{P1\text{-}P3,h}) \rfloor \quad \text{(C.47)}$$

C.5.3. Bound on the observed relative skew of received messages for process P3

Let $\Pi_{P3,RCV}$ denote the bound on the relative skew observed in process P3 for the received messages from trustworthy sources in process P2. $\Pi_{P3,RCV}$ is measured in local clock ticks. $\Pi_{P3,RCV}$ is used to check for agreement among the received inputs and also to check agreement with the result of the Accept output.

The worst case relative skew for received messages occurs when there are asymmetric eligible voters in process P2 at the trustworthy BIU nodes. The bound on the relative skew of the Accept outputs in process P2 is $\pi_{P2,A}$. The additional uncertainty in the reception delay measured from the time of the Accept outputs in process P2 to the time of reception in process P3 is $e_{PP} + [(1+\rho_0) - 1/(1+\rho_0)]B_{P2}$.

$$\begin{aligned}\Pi_{P3,RCV} &= \lfloor (1 + \rho_0)(t_{P3,RCV,h} - t_{P3,RCV,l}) \rfloor \\ &= \lfloor (1 + \rho_0)\{\pi_{P2,A} + e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)]B_{P2}\} \rfloor \\ &= \lfloor (1 + \rho_0)\{3e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](A_{P1} + B_{P1} + A_{P2} + B_{P2})\} \rfloor \end{aligned} \quad \text{(C.48)}$$

C.5.4. Relative skew of the Accept outputs for process P3

Let AEV_P3 denote the set of asymmetric BIU eligible voters in process P3 at a trustworthy RMU node. Let $\pi_{P3,A}$ denote the bound on the real-time relative skew of the Accept outputs in process P3 at the trustworthy RMUs. A_{P3} denotes the delay (in local-clock ticks) of the Computation Process in process P3 measured from the local time of reception of the selected message to the local time when the Accept output is asserted. If $|AEV\_P2| = 0$ for each trustworthy BIU node, the trustworthy RMU nodes may have asymmetric BIU nodes in their sets of eligible voters for process P3 (i.e., $|AEV\_P3| \neq 0$ for some trustworthy RMU nodes). In this case, the trustworthy RMU nodes accept within the time range delimited by messages from trustworthy BIU nodes.

$$\pi_{P3,A}|_{|AEV\_P3| \neq 0} = \pi_{P2,A}|_{|AEV\_P2| = 0} + e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](B_{P2} + A_{P3})$$

$$= 2e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](A_{P2} + B_{P2} + A_{P3}) \quad \text{(C.49)}$$

If |AEV_P2| ≠ 0 for some trustworthy BIU nodes, the trustworthy RMU nodes do not have asymmetric BIU nodes in their sets of eligible voters for process P3 (i.e., |AEV_P3| = 0 for each trustworthy RMU node). In this case, the RMU nodes essentially accept on the same message.

$$\pi_{P3,A}|_{|AEV\_P3| = 0} = e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)]A_{P3} \quad \text{(C.50)}$$

From this point on, unless otherwise stated:

$$\pi_{P3,A} = \pi_{P3,A}|_{max} = \pi_{P3,A}|_{|AEV\_P3| \neq 0} \quad \text{(C.51)}$$

C.6. Fourth stage

Figure C.7 illustrates the detailed message flow graph up to stage 4 for a 3x3 system.

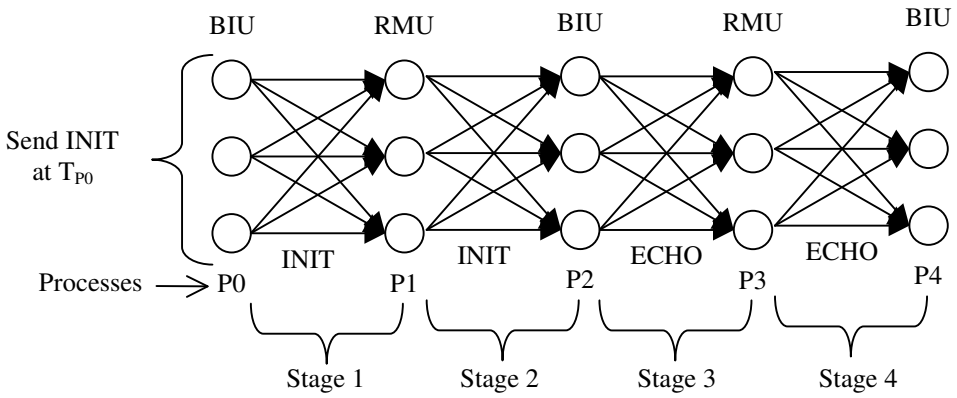


Figure C.7: Detailed message flow graph for stages 1 through 4 in a 3x3 system

C.6.1. Effective reception delay for process P4

We need to determine the earliest and latest real time of reception of ECHO messages from trustworthy RMU nodes by the BIU nodes in process P4. Let $T_{P2,A,j}$ denote the local time at which trustworthy BIU node j asserts the output of its Accept function for process P2. $t_{P2,A,l}$ and $t_{P2,A,h}$ denote the earliest and latest real times, respectively, at which the trustworthy BIUs can assert their Accept outputs in process P2.

$$\pi_{P2,A} = t_{P2,A,h} - t_{P2,A,l} \quad \text{(C.52)}$$

Let B_{P3} denote the send delay for process P3. $t_{P4,RCV,l}$ denotes the earliest real time at which ECHO messages from trustworthy RMU nodes can be received by a BIU node in process P4.

$$t_{P4,RCV,l} = t_{P2,A,l} + 2r_{PP,l} + (B_{P2} + A_{P3} + B_{P3})/(1 + \rho_0) \quad \text{(C.53)}$$

$t_{P4,RCV,h}$ denotes the latest real time at which ECHO messages from trustworthy RMU nodes can

be received by a BIU node in process P4.

$$t_{P4,RCV,h} = t_{P2,A,h} + 2r_{PP,h} + (1 + \rho_0)(B_{P2} + A_{P3} + B_{P3}) \quad (C.54)$$

$r_{P2\text{-}P4,l}$ denotes the minimum effective message-reception delay for ECHO messages in process P4 and is measured from the latest time at which the trustworthy BIU nodes can assert their Accept(ECHO) outputs to the earliest time at which the BIU nodes can receive ECHO messages from the trustworthy RMU nodes.

$$r_{P2\text{-}P4,l} = t_{P4,RCV,l} - t_{P2,A,h} = 2r_{PP,l} + (B_{P2} + A_{P3} + B_{P3})/(1 + \rho_0) - \pi_{P2,A} \quad (C.55)$$

$r_{P2\text{-}P4,h}$ denotes the maximum effective message-reception delay for ECHO messages in process P4 and is measured from the earliest time at which the trustworthy BIU nodes assert their Accept(ECHO) outputs to the latest time at which the BIU nodes can receive ECHO messages from the trustworthy RMU nodes.

$$r_{P2\text{-}P4,h} = t_{P4,RCV,h} - t_{P2,A,l} = \pi_{P2,A} + 2r_{PP,h} + (1 + \rho_0)(B_{P2} + A_{P3} + B_{P3}) \quad (C.56)$$

The expected reception delay for process P4 is:

$$R_{P2\text{-}P4} = IMP(r_{P2\text{-}P4,l}\ ,\ r_{P2\text{-}P4,h}) \quad (C.57)$$

The effective uncertainty in the real time of reception of the ECHO messages in process P4 is:

$$r_{P2\text{-}P4,h} - r_{P2\text{-}P4,l} = 2\pi_{P2,A} + 2e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](B_{P2} + A_{P3} + B_{P3}) \quad (C.58)$$

C.6.2. Expected time of reception for process P4

BIU node j expect to receive ECHO messages at local time $T_{P4,RCV,E,j}$:

$$T_{P4,RCV,E,j} = T_{P2,A,j} + R_{P2\text{-}P4} \quad (C.59)$$

The real-time error for $T_{P4,RCV,E,j}$ is bounded as follows. BIU node j will receive an ECHO message from a trustworthy RMU node no earlier than $\mu_{P2\text{-}P4,l}$ nominal ticks from $T_{P4,RCV,E,j}$:

$$\mu_{P2\text{-}P4,l} = (1 + \rho_0)R_{P2\text{-}P4} - r_{P2\text{-}P4,l} \quad (C.60)$$

BIU node j will receive an ECHO message from a trustworthy RMU node no later than $\mu_{P2\text{-}P4,h}$ nominal ticks from $T_{P4,RCV,E,j}$:

$$\mu_{P2\text{-}P4,h} = r_{P2\text{-}P4,h} - R_{P2\text{-}P4}/(1 + \rho_0) \quad (C.61)$$

We want to determine the maximum local-time error for the actual time of reception at the BIU nodes in process P4, denoted by $\Delta_{P4,RCV}|_{max}$.

$$|T_{P4,RCV} - T_{P4,RCV,E}| \leq \Delta_{P4,RCV}|_{max} \quad (C.62)$$

Following the analysis for process P2:

$$\Delta_{P4,RCV}|_{max} = \lfloor (1 + \rho_0)\max(\mu_{P2\text{-}P4,l}\ ,\ \mu_{P2\text{-}P4,h}) \rfloor \qquad \text{(C.63)}$$

C.6.3. Bound on the observed relative skew of received messages for process P4

Let $\Pi_{P4,RCV}$ denote the bound on the relative skew observed in process P4 for the received messages from process P3 at trustworthy RMUs. $\Pi_{P4,RCV}$ is measured in local clock ticks. $\Pi_{P4,RCV}$ is used to check for agreement among the received inputs and also to check agreement with the result of the Accept output.

The worst case relative skew for received messages occurs when there are asymmetric eligible voters in process P3 at trustworthy RMU nodes. The bound on the relative skew of the Accept outputs in process P3 at the trustworthy RMUs is $\pi_{P3,A}$. The additional uncertainty in the reception delay measured from the time of the Accept outputs in process P3 to the time of reception in process P4 is $e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)]B_{P3}$.

$$\begin{aligned}\Pi_{P4,RCV} &= \lfloor (1 + \rho_0)(t_{P4,RCV,h} - t_{P4,RCV,l}) \rfloor \\ &= \lfloor (1 + \rho_0)\{\pi_{P3,A} + e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)]B_{P3}\} \rfloor \\ &= \lfloor (1 + \rho_0)\{3e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](A_{P2} + B_{P2} + A_{P3} + B_{P3})\} \rfloor \end{aligned} \qquad \text{(C.64)}$$

C.6.4. Relative skew of the Accept outputs for process P4

Let AEV_P4 denote the set of asymmetric RMU eligible voters in process P4 at a trustworthy BIU node. Let $\pi_{P4,A}$ denote the bound on the real-time relative skew of the Accept outputs in process P4 at the trustworthy BIUs. A_{P4} denotes the delay (in local-clock ticks) of the Computation Process in process P4 measured from the local time of reception of the selected message to the local time when the Accept output is asserted. If |AEV_P3| = 0 for each trustworthy RMU node, the trustworthy BIU nodes may have asymmetric RMU nodes in their sets of eligible voters for process P4 (i.e., $|AEV\_P4| \neq 0$ for some trustworthy BIU nodes). In this case, the BIU nodes accept within the time range delimited by messages from trustworthy RMU nodes.

$$\begin{aligned}\pi_{P4,A}|_{|AEV\_P4| \neq 0} &= \pi_{P3,A}|_{|AEV\_P3| = 0} + e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](B_{P3} + A_{P4}) \\ &= 2e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](A_{P3} + B_{P3} + A_{P4}) \end{aligned} \qquad \text{(C.65)}$$

If $|AEV\_P3| \neq 0$ for some trustworthy RMU nodes, the trustworthy BIU nodes do not have asymmetric RMU nodes in their sets of eligible voters for process P4 (i.e., |AEV_P4| = 0 for each trustworthy BIU node). In this case, the BIU nodes essentially accept on the same message.

$$\pi_{P4,A}|_{|AEV\_P4| = 0} = e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)]A_{P4} \qquad \text{(C.66)}$$

From this point on, unless otherwise stated:

$$\pi_{P4,A} = \pi_{P4,A}|_{max} = \pi_{P4,A}|_{|AEV\_P4| \neq 0} \qquad \text{(C.67)}$$

C.7. Synchronization capture

Figure C.8 illustrates the detailed message flow graph for the synchronization-capture stages in a 3x3 system. The nodes executing processes P3C and P4C are called **recovering** nodes.

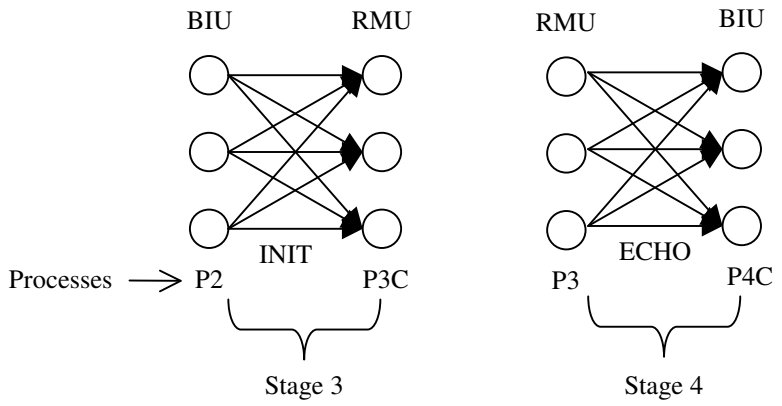


Figure C.8: Detailed message flow graph for synchronization-capture stages in a 3x3 system

C.7.1. Bound on the observed relative skew of received messages for process P3C

Let $\Pi_{P3C,RCV}$ denote the bound on the relative skew observed in process P3C for the received messages from process P2 at trustworthy BIUs. $\Pi_{P3C,RCV}$ is measured in local clock ticks. $\Pi_{P3C,RCV}$ is used to check for agreement among the received inputs and also to check agreement with the result of the Accept output.

The nodes in process P3C receive ECHO messages from process P2 at trustworthy BIUs in the same real time range as nodes executing process P3. Therefore:

$$\Pi_{P3C,RCV} = \Pi_{P3,RCV} \tag{C.68}$$

C.7.2. Relative skew of the Accept outputs for process P3C

Recovering RMU nodes synchronize using the ECHO messages from process P2 of the Synchronization Preservation protocol. Because the recovering nodes may have asymmetric faulty nodes in their sets of eligible voters, all we know is that they will accept within the time range delimited by ECHO messages from trustworthy BIU nodes. Let $\pi_{P3C,A}$ denote the bound on the real-time relative skew of the Accept outputs in process P3C at the good recovering RMUs. Let A_{P3C} denote the delay (in local-clock ticks) of the Computation Process in process P3C measured from the local time of reception of the selected message to the local time when the Accept output is asserted. We assume that the delay of the Computation Process in process P3C is the same as in process P3.

$$A_{P3C} = A_{P3} \tag{C.69}$$

The worst-case real-time relative skew occurs when the trustworthy BIU nodes and the

recovering RMU nodes simultaneously have asymmetric nodes in their sets of eligible voters. For that case:

$$\pi_{P3C,A} = \pi_{P2,A} + e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](B_{P2} + A_{P3C})$$

$$= 3e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](A_{P1} + B_{P1} + A_{P2} + B_{P2} + A_{P3}) \quad \text{(C.70)}$$

C.7.3. Bound on the observed relative skew of received messages for process P4C

Let $\Pi_{P4C,RCV}$ denote the bound on the maximum relative skew observed in process P4C for the received messages from process P3 at trustworthy RMUs. $\Pi_{P4C,RCV}$ is measured in local clock ticks. $\Pi_{P4C,RCV}$ is used to check for agreement among the received inputs and also to check agreement with the result of the Accept output.

The nodes executing process P4C receive ECHO messages from process P3 at trustworthy RMUs in the same time range as nodes executing process P4. Therefore:

$$\Pi_{P4C,RCV} = \Pi_{P4,RCV} \quad \text{(C.71)}$$

C.7.4. Relative skew of the Accept outputs for process P4C

Recovering BIU nodes synchronize using the ECHO messages from process P3 of the Synchronization Preservation protocol. Because the recovering BIUs may have asymmetric faulty nodes in their sets of eligible voters, all we know is that they will accept within the time range delimited by ECHO messages from trustworthy RMU nodes. Let $\pi_{P4C,A}$ denote the bound on the real-time relative skew of the Accept outputs in process P4C at the good recovering BIUs. Let A_{P4C} denote the delay (in local-clock ticks) of the Computation Process in process P4C measured from the local time of reception of the selected message to the local time when the Accept output is asserted. We assume that the delay of the Computation Process in process P4C is the same as in process P4.

$$A_{P4C} = A_{P4} \quad \text{(C.72)}$$

The worst-case real-time relative skew occurs when the trustworthy RMU nodes and the recovering BIU nodes simultaneously have asymmetric nodes in their sets of eligible voters. For that case:

$$\pi_{P4C,A} = \pi_{P3,A} + e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](B_{P3} + A_{P4C})$$

$$= 3e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](A_{P2} + B_{P2} + A_{P3} + B_{P3} + A_{P4}) \quad \text{(C.73)}$$

C.8. Resetting the local time

C.8.1. Relative skew of the local-time reset for process P4

Let $T_{P4,A,j}$ denote the local time of the Accept output in process P4 at trustworthy BIU j. H_{P4}

denotes the synchronization-reset delay applied by the BIU nodes resetting with respect to the Accept output in process P4. $T_{P4,H,j}$ denotes the local time at which the next cycle begins for BIU node j synchronizing with respect to process P4.

$$T_{P4,H,j} = T_{P4,A,j} + H_{P4} \tag{C.74}$$

$\pi_{P4,H}$ denotes the bound on the relative skew of the local-time reset for BIU nodes synchronizing with respect to process P4. Then:

$$\pi_{P4,H} = \pi_{P4,A} + [(1 + \rho_0) - 1/(1 + \rho_0)]H_{P4}$$

$$= 2e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](A_{P3} + B_{P3} + A_{P4} + H_{P4}) \tag{C.75}$$

C.8.2. Relative skew of the local-time reset for process P4C

Let $T_{P4C,A,j}$ denote the local time of the Accept output in process P4C at a good recovering BIU j. H_{P4C} denotes the synchronization-reset delay applied by the nodes resetting with respect to the Accept output in process P4C at the good recovering BIUs. $T_{P4C,H,j}$ denotes the local time at which the next cycle begins for BIU node j synchronizing with respect to process P4C. The BIU nodes executing process P4C apply the same synchronization-reset delay as the nodes executing process P4.

$$H_{P4C} = H_{P4} \tag{C.76}$$

So:

$$T_{P4C,H,j} = T_{P4C,A,j} + H_{P4C} = T_{P4C,A,j} + H_{P4} \tag{C.77}$$

The bound on the relative skew of the Accept output for process P4C at the good recovering BIUs is given by $\pi_{P4C,A}$. $\pi_{P4C,H}$ denotes the bound on the relative skew of the local-time reset for good recovering BIU nodes synchronizing with respect to process P4C. Then:

$$\pi_{P4C,H} = \pi_{P4C,A} + [(1 + \rho_0) - 1/(1 + \rho_0)]H_{P4}$$

$$= 3e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](A_{P2} + B_{P2} + A_{P3} + B_{P3} + A_{P4} + H_{P4}) \tag{C.78}$$

C.8.3. Reset delay for process P3

Let $T_{P3,A,i}$ denote the local time of the Accept output in process P3 at trustworthy RMU i. H_{P3} denotes the synchronization-reset delay applied by the nodes resetting with respect to the Accept output in process P3. $T_{P3,H,i}$ denotes the local time at which the next cycle begins for RMU i synchronizing with respect to process P3.

$$T_{P3,H,i} = T_{P3,A,i} + H_{P3} \tag{C.79}$$

H_{P3} is the expected delay from the time when the RMU nodes in process P3 assert their Accept output until the BIU nodes synchronizing with respect to process P4 reset their local-time clocks. The bound on the relative skew of the Accept outputs in process P3 at the trustworthy RMUs is

given by $\pi_{P3,A}$. $t_{P3,A,l}$ and $t_{P3,A,h}$ denote the earliest and latest real times, respectively, at which the Accept outputs can be asserted in process P3 at the trustworthy RMUs. So:

$$\pi_{P3,A} = t_{P3,A,h} - t_{P3,A,l} \tag{C.80}$$

$t_{P4,H,l}|_{P3,A}$ and $t_{P4,H,h}|_{P3,A}$ denote the earliest and latest real times, respectively, at which a trustworthy BIU node synchronizing with respect to process P4 can reset its local-time clock. $t_{P4,H,l}|_{P3,A}$ and $t_{P4,H,h}|_{P3,A}$ are measured with respect to the Accept outputs in process P3 at the trustworthy RMUs.

$$t_{P4,H,l}|_{P3,A} = t_{P3,A,l} + r_{PP,l} + (B_{P3} + A_{P4} + H_{P4})/(1 + \rho_0) \tag{C.81}$$

$$t_{P4,H,h}|_{P3,A} = t_{P3,A,h} + r_{PP,h} + (1 + \rho_0)(B_{P3} + A_{P4} + H_{P4}) \tag{C.82}$$

Let $h_{P3,l}$ denote the minimum effective delay from the time the Accept output in process P3 at trustworthy RMUs is asserted to the time a trustworthy BIU node resets its local-time clock with respect to process P4.

$$\begin{aligned} h_{P3,l} &= t_{P4,H,l}|_{P3,A} - t_{P3,A,h} \\ &= r_{PP,l} + (B_{P3} + A_{P4} + H_{P4})/(1 + \rho_0) - \pi_{P3,A} \end{aligned} \tag{C.83}$$

$h_{P3,h}$ denotes the maximum effective delay from the time the Accept output in process P3 at trustworthy RMUs is asserted to the time a trustworthy BIU node resets its local-time clock with respect to process P4.

$$\begin{aligned} h_{P3,h} &= t_{P4,H,h}|_{P3,A} - t_{P3,A,l} \\ &= \pi_{P3,A} + r_{PP,h} + (B_{P3} + A_{P4} + H_{P4})(1 + \rho_0) \end{aligned} \tag{C.84}$$

H_{P3} is given by:

$$H_{P3} = IMP(h_{P3,l}, h_{P3,h}) \tag{C.85}$$

The real-time error for $T_{P3,H,i}$ is bounded as follows. A trustworthy BIU node can reset its local-time clock with respect to process P4 no earlier than $\mu_{P3,H,l}$ nominal ticks from local time $T_{P3,H,i}$ at a trustworthy RMU node synchronizing with respect to process P3.

$$\mu_{P3,H,l} = (1 + \rho_0)H_{P3} - h_{P3,l} \tag{C.86}$$

A trustworthy BIU node can reset its local-time clock with respect to process P4 no later than $\mu_{P3,H,h}$ nominal ticks from local time $T_{P3,H,i}$ at a trustworthy RMU node synchronizing with respect to process P3.

$$\mu_{P3,H,h} = h_{P3,h} - H_{P3}/(1 + \rho_0) \tag{C.87}$$

Note that this analysis also applies to the real-time error for $T_{P3,H,i}$ with respect to the local-time reset of nodes synchronizing in process P4C.

C.8.4. Relative skew of the local-time reset between processes P3, and P4 or P4C

Let $\pi_{P3\text{-}P4,H}$ denote the bound on the relative skew of the local-time reset between RMU nodes synchronizing with respect to process P3 and BIU nodes synchronizing with respect to process P4.

$$\pi_{P3\text{-}P4,H} = \max(\mu_{P3,H,l}, \mu_{P3,H,h}) \tag{C.88}$$

$\pi_{P3\text{-}P4C,H}$ denotes the bound on the relative skew of the local-time reset between RMU nodes synchronizing with respect to process P3 and BIU nodes synchronizing with respect to process P4C. $\pi_{P3\text{-}P4,H}$ also applies here.

$$\pi_{P3\text{-}P4C,H} = \pi_{P3\text{-}P4,H} \tag{C.89}$$

C.8.5. Relative skew of the local-time reset for process P3

The bound on the relative skew of the Accept outputs in process P3 at trustworthy RMUs is given by $\pi_{P3,A}$. $\pi_{P3,H}$ denotes the bound on the relative skew of the local-time reset for trustworthy RMU nodes resetting with respect to the Accept output in process P3.

$$\pi_{P3,H} = \pi_{P3,A} + [(1 + \rho_0) - 1/(1 + \rho_0)]H_{P3}$$

$$= 2e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](A_{P2} + B_{P2} + A_{P3} + H_{P3}) \tag{C.90}$$

C.8.6. Relative skew of the local-time reset for process P3C

Let $T_{P3C,A,i}$ denote the local time of the Accept output in process P3C at good recovering RMU i. H_{P3C} denotes the synchronization-reset delay applied by the RMU nodes resetting with respect to the Accept output in process P3C. $T_{P3C,H,i}$ denotes the local time at which the next cycle begins for RMU node i synchronizing with respect to process P3C. The RMU nodes executing process P3C apply the same synchronization-reset delay as the RMU nodes executing process P3.

$$H_{P3C} = H_{P3} \tag{C.91}$$

So:

$$T_{P3C,H,i} = T_{P3C,A,i} + H_{P3C} = T_{P3C,A,i} + H_{P3} \tag{C.92}$$

The bound on the relative skew for the Accept outputs in process P3C at the good recovering RMUs is given by $\pi_{P3C,A}$. $\pi_{P3C,H}$ denotes the bound on the relative skew of the local-time reset for the nodes synchronizing with respect to the Accept output in process P3C.

$$\pi_{P3C,H} = \pi_{P3C,A} + [(1 + \rho_0) - 1/(1 + \rho_0)]H_{P3}$$

$$= 3e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](A_{P1} + B_{P1} + A_{P2} + B_{P2} + A_{P3} + H_{P3}) \tag{C.93}$$

C.8.7. Reset delay for process P2

Let $T_{P2,A,k}$ denote the local time of the Accept output in process P2 at trustworthy BIU k. H_{P2} denotes the synchronization-reset delay applied by the BIU nodes resetting with respect to the Accept output in process P2. $T_{P2,H,k}$ denotes the local time at which the next cycle begins for BIU node k synchronizing with respect to process P2.

$$T_{P2,H,k} = T_{P2,A,k} + H_{P2} \tag{C.94}$$

H_{P2} is the expected delay from the time when the BIU nodes executing process P2 assert their Accept outputs until the RMU nodes synchronizing with respect to process P3 reset their local-time clocks. $t_{P3,H,l|P2,A}$ denotes the earliest real time at which a trustworthy RMU node synchronizing with respect to process P3 can reset its local-time clock, measured with respect to the Accept outputs in process P2 at the trustworthy BIUs.

$$t_{P3,H,l|P2,A} = t_{P2,A,l} + r_{PP,l} + (B_{P2} + A_{P3} + H_{P3})/(1 + \rho_0) \tag{C.95}$$

Let $t_{P3,H,h|P2,A}$ denote the latest real time at which a trustworthy RMU node synchronizing with respect to process P3 can reset its local-time clock, measured with respect to the Accept outputs in process P2 at the trustworthy BIUs.

$$t_{P3,H,h|P2,A} = t_{P2,A,h} + r_{PP,h} + (B_{P2} + A_{P3} + H_{P3})(1 + \rho_0) \tag{C.96}$$

Let $h_{P2,l}$ denote the minimum effective delay from the time a trustworthy BIU node in process P2 asserts its Accept output until a trustworthy RMU node in process P3 resets its local-time clock.

$$\begin{aligned} h_{P2,l} &= t_{P3,H,l|P2,A} - t_{P2,A,h} \\ &= r_{PP,l} + (B_{P2} + A_{P3} + H_{P3})/(1 + \rho_0) - \pi_{P2,A} \end{aligned} \tag{C.97}$$

Let $h_{P2,h}$ denote the maximum effective delay from the time a trustworthy BIU node in process P2 asserts its Accept output until a trustworthy RMU node in process P3 resets its local-time clock.

$$\begin{aligned} h_{P2,h} &= t_{P3,H,h|P2,A} - t_{P2,A,l} \\ &= \pi_{P2,A} + r_{PP,h} + (B_{P2} + A_{P3} + H_{P3})(1 + \rho_0) \end{aligned} \tag{C.98}$$

H_{P2} is given by:

$$H_{P2} = IMP(h_{P2,l}, h_{P2,h}) \tag{C.99}$$

The real-time error for $T_{P2,H,k}$ is bounded as follows. A trustworthy RMU node in process P3 can reset its local-time clock no earlier than $\mu_{P2,H,l}$ nominal ticks from local time $T_{P2,H,k}$ at a BIU node synchronizing with respect to process P2.

$$\mu_{P2,H,l} = (1 + \rho_0)H_{P2} - h_{P2,l} \tag{C.100}$$

A trustworthy RMU node in process P3 can reset its local-time clock no later than $\mu_{P2,H,h}$ nominal ticks from local time $T_{P2,H,k}$ at a BIU node synchronizing with respect to process P2.

$$\mu_{P2,H,h} = h_{P2,h} - H_{P2}/(1 + \rho_0) \quad \text{(C.101)}$$

Note that this analysis also applies to the real-time error for $T_{P2,H,k}$ with respect to the local-time reset of nodes synchronizing with respect to process P3C.

C.8.8. Relative skew of the local-time reset between processes P2, and P3 or P3C

Let $\pi_{P2\text{-}P3,H}$ denote the bound on the relative skew of the local-time reset between trustworthy BIU nodes synchronizing with respect to process P2 and trustworthy RMU nodes synchronizing with respect to process P3.

$$\pi_{P2\text{-}P3,H} = \max(\mu_{P2,H,l}\ ,\ \mu_{P2,H,h}) \quad \text{(C.102)}$$

$\pi_{P2\text{-}P3C,H}$ denotes the bound on the relative skew of the local-time reset between trustworthy BIU nodes synchronized with respect to process P2 and good recovering RMU nodes synchronized with respect to process P3C. $\pi_{P2\text{-}P3,H}$ also applies here.

$$\pi_{P2\text{-}P3C,H} = \pi_{P2\text{-}P3,H} \quad \text{(C.103)}$$

C.8.9. Relative skew of the local-time reset for process P2

The bound on the relative skew of the Accept outputs in process P2 at trustworthy BIUs is given by $\pi_{P2,A}$. $\pi_{P2,H}$ denotes the bound on the relative skew of the local-time reset for trustworthy BIU nodes synchronizing with respect to process P2. Then:

$$\pi_{P2,H} = \pi_{P2,A} + [(1 + \rho_0) - 1/(1 + \rho_0)]H_{P2}$$

$$= 2e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)](A_{P1} + B_{P1} + A_{P2} + H_{P2}) \quad \text{(C.104)}$$

C.8.10. Relative skew of the local-time reset for a set including processes P2 and P3C

Let $t_{P3C,H,l|P2,A}$ denote the earliest real time at which a good recovering RMU node synchronizing with respect to process P3C can reset its local-time clock, measured with respect to the Accept outputs in process P2 at trustworthy BIUs.

$$t_{P3C,H,l|P2,A} = t_{P2,A,l} + r_{PP,l} + (B_{P2} + A_{P3} + H_{P3})/(1 + \rho_0) \quad \text{(C.105)}$$

$t_{P3C,H,h|P2,A}$ denotes the latest real time at which a good recovering RMU node synchronizing with respect to process P3C can reset its local-time clock, measured with respect to the Accept outputs in process P2 at trustworthy BIUs.

$$t_{P3C,H,h|P2,A} = t_{P2,A,h} + r_{PP,h} + (1 + \rho_0)(B_{P2} + A_{P3} + H_{P3}) \quad \text{(C.106)}$$

$t_{P2,H,l}$ denotes the earliest real time at which a trustworthy BIU node synchronizing with respect to process P2 can reset its local-time clock.

$$t_{P2,H,l} = t_{P2,A,l} + H_{P2}/(1 + \rho_0) \quad \text{(C.107)}$$

$t_{P2,H,h}$ denotes the latest real time at which a trustworthy BIU node synchronizing with respect to process P2 can reset its local-time clock.

$$t_{P2,H,h} = t_{P2,A,h} + H_{P2}(1 + \rho_0) \quad \text{(C.108)}$$

$\pi_{P2+P3C,H}$ denotes the bound on the relative skew of the local-time reset for a node set including all the trustworthy or good recovering nodes synchronizing with respect to process P2 or P3C.

$$\pi_{P2+P3C,H} = \max(|\, t_{P3C,H,h}|_{P2,A} - t_{P2,H,l}\,|, |\, t_{P2,H,h} - t_{P3C,H,l}|_{P2,A}\,|, \pi_{P2,H}, \pi_{P3C,H}) \quad \text{(C.109)}$$

C.8.11. Relative skew of the local-time reset for a set including processes P2 and P3

Let $\pi_{P2+P3,H}$ denote the bound on the relative skew of the local-time reset for a node set including all trustworthy BIU nodes synchronizing with respect to process P2 or P3. With respect to process P2, the Accept outputs in process P3 at the trustworthy RMUs and the Accept outputs in process P3C at the good recovering RMUs can be asserted during the same real time interval. In the presence of asymmetric faulty BIU nodes, we know that the time interval of the Accept outputs in process P3 at the trustworthy RMUs is contained within the time interval of the Accept outputs in process P3C at the good recovering RMUs. Therefore:

$$\pi_{P2+P3,H} \leq \pi_{P2+P3C,H} \quad \text{(C.110)}$$

C.8.12. Relative skew of the local-time reset for a set including processes P2 and P4C

Let $t_{P4C,H,l}|_{P2,A}$ denote the earliest real time at which a good recovering BIU node synchronizing with respect to process P4C can reset its local-time clock, measured with respect to the Accept outputs in process P2 at the trustworthy BIUs.

$$t_{P4C,H,l}|_{P2,A} = t_{P2,A,l} + 2r_{PP,l} + (B_{P2} + A_{P3} + B_{P3} + A_{P4} + H_{P4})/(1 + \rho_0) \quad \text{(C.111)}$$

$t_{P4C,H,h}|_{P2,A}$ denotes the latest real time at which a good recovering BIU node synchronizing with respect to process P4C can reset its local-time clock, measured with respect to the Accept outputs in process P2 at the trustworthy BIUs.

$$t_{P4C,H,h}|_{P2,A} = t_{P2,A,h} + 2r_{PP,h} + (1 + \rho_0)(B_{P2} + A_{P3} + B_{P3} + A_{P4} + H_{P4}) \quad \text{(C.112)}$$

$\pi_{P2+P4C,H}$ denotes the bound on the relative skew of the local-time reset for a node set including all BIU nodes synchronizing with respect to process P2 or P4C.

$$\pi_{P2+P4C,H} = \max(|\, t_{P4C,H,h}|_{P2,A} - t_{P2,H,l}\,|, |\, t_{P2,H,h} - t_{P4C,H,l}|_{P2,A}\,|, \pi_{P2,H}, \pi_{P4C,H}) \quad \text{(C.113)}$$

C.8.13. Relative skew of the local-time reset for a set including processes P2 and P4

Let $\pi_{P2+P4,H}$ denote the bound on the relative skew of the local-time reset for a node set including all trustworthy BIU nodes synchronizing with respect to process P2 or P4. With respect to process P2, the Accept outputs in process P4 at the trustworthy BIUs and the Accept outputs in process P4C at the good recovering BIUs can be asserted during the same real time interval. In

the presence of asymmetric faulty RMU nodes, we know that the time interval of the Accept outputs in process P4 at the trustworthy BIUs is contained within the time interval of the Accept outputs in process P4C at the good recovering BIUs. Therefore:

$$\pi_{P2+P4,H} \leq \pi_{P2+P4C,H} \tag{C.114}$$

C.8.14. Relative skew of the local-time reset for a set including processes P3 or P3C

Let $\pi_{P3+P3C,H}$ denote the bound on the relative skew of the local-time reset for a node set including all trustworthy or good recovering RMU nodes synchronizing with respect to process P3 or P3C. With respect to process P2, the Accept outputs in process P3 at the trustworthy RMUs and the Accept outputs in process P3C at the good recovering RMUs can be asserted during the same real time interval. This interval is determined by the time range during which the trustworthy BIU nodes executing process P2 send ECHO. In the presence of asymmetric faulty BIU nodes, the good recovering RMU nodes executing process P3C may not be able to synchronize any better than the duration of this time range.

$$\pi_{P3+P3C,H} = \max(\pi_{P3,H}, \pi_{P3C,H}) = \pi_{P3C,H} \tag{C.115}$$

C.8.15. Relative skew of the local-time reset for a set including processes P3 and P4C

Let $t_{P4C,H,l|P3,A}$ denote the earliest real time at which a good recovering BIU node synchronizing with respect to process P4C can reset its local-time clock, measured with respect to the Accept outputs in process P3 at the trustworthy RMUs.

$$t_{P4C,H,l|P3,A} = t_{P3,A,l} + r_{PP,l} + (B_{P3} + A_{P4} + H_{P4})/(1 + \rho_0) \tag{C.116}$$

$t_{P4C,H,h|P3,A}$ denotes the latest real time at which a good recovering BIU node synchronizing with respect to process P4C can reset its local-time clock, measured with respect to the Accept outputs in process P3 at the trustworthy RMUs.

$$t_{P4C,H,h|P3,A} = t_{P3,A,h} + r_{PP,h} + (1 + \rho_0)(B_{P3} + A_{P4} + H_{P4}) \tag{C.117}$$

$t_{P3,H,l|P3,A}$ denotes the earliest real time at which a trustworthy RMU node synchronizing with respect to process P3 can reset its local-time clock, measured with respect to the Accept outputs in process P3 at the trustworthy RMUs.

$$t_{P3,H,l|P3,A} = t_{P3,A,l} + H_{P3}/(1 + \rho_0) \tag{C.118}$$

$t_{P3,H,h|P3,A}$ denotes the latest real time at which a trustworthy RMU node synchronizing with respect to process P3 can reset its local-time clock, measured with respect to the Accept outputs in process P3 at the trustworthy RMUs.

$$t_{P3,H,h|P3,A} = t_{P3,A,h} + H_{P3}(1 + \rho_0) \tag{C.119}$$

$\pi_{P3+P4C,H}$ denotes the bound on the relative skew of the local-time reset for a node set including all trustworthy or good recovering nodes synchronizing with respect to process P3 or P4C.

$$\pi_{P3+P4C,H} = \max(|\, t_{P4C,H,h}|_{P3,A} - t_{P3,H,l}|_{P3,A}\,|, |\, t_{P3,H,h}|_{P3,A} - t_{P4C,H,l}|_{P3,A}\,|, \pi_{P3,H}, \pi_{P4C,H}) \quad \text{(C.120)}$$

C.8.16. Relative skew of the local-time reset for a set including processes P3 and P4

Let $\pi_{P3+P4,H}$ denote the bound on the relative skew of the local-time reset for a node set including all BIU nodes synchronizing with respect to process P3 or P4. With respect to process P3, the Accept outputs in process P4 at the trustworthy BIUs and the Accept outputs in process P4C at the good recovering BIUs can be asserted during the same real time interval. In the presence of asymmetric faulty RMU nodes, we know that the time interval of the Accept outputs in process P4 at the trustworthy BIU nodes is contained within the time interval of the Accept outputs in process P4C at the good recovering BIU nodes. Therefore:

$$\pi_{P3+P4,H} \leq \pi_{P3+P4C,H} \quad \text{(C.121)}$$

C.8.17. Relative skew of the local-time reset for a set including processes P3C and P4C

Let $t_{P3C,H,l}|_{P2,A}$ denote the earliest real time at which a good recovering RMU node synchronizing with respect to process P3C can reset its local-time clock, measured with respect to the Accept outputs in process P2 at the trustworthy BIUs.

$$t_{P3C,H,l}|_{P2,A} = t_{P2,A,l} + r_{PP,l} + (B_{P2} + A_{P3} + H_{P3})/(1 + \rho_0) \quad \text{(C.122)}$$

$t_{P3C,H,h}|_{P2,A}$ denotes the latest real time at which a good recovering RMU node synchronizing with respect to process P3C resets its local-time clock, measured with respect to the Accept outputs in process P2.

$$t_{P3C,H,h}|_{P2,A} = t_{P2,A,h} + r_{PP,h} + (1 + \rho_0)(B_{P2} + A_{P3} + H_{P3}) \quad \text{(C.123)}$$

$t_{P4C,H,l}|_{P2,A}$ denotes the earliest real time at which a good BIU node synchronizing with respect to process P4C can reset its local-time clock, measured with respect to the Accept outputs in process P2 at the trustworthy BIUs.

$$t_{P4C,H,l}|_{P2,A} = t_{P2,A,l} + 2r_{PP,l} + (B_{P2} + A_{P3} + B_{P3} + A_{P4} + H_{P4})/(1 + \rho_0) \quad \text{(C.124)}$$

$t_{P4C,H,h}|_{P2,A}$ denotes the latest real time at which a good recovering BIU node synchronizing with respect to process P4C can reset its local-time clock, measured with respect to the Accept outputs in process P2 at the trustworthy BIUs.

$$t_{P4C,H,h}|_{P2,A} = t_{P2,A,h} + 2r_{PP,h} + (1 + \rho_0)(B_{P2} + A_{P3} + B_{P3} + A_{P4} + H_{P4}) \quad \text{(C.125)}$$

$\pi_{P3C+P4C,H}$ denotes the bound on the relative skew of the local-time reset for a node set including all good recovering nodes synchronizing with respect to process P3C or P4C.

$$\pi_{P3C+P4C,H} = \max(|\, t_{P4C,H,h}|_{P2,A} - t_{P3C,H,l}|_{P2,A}\,|, |\, t_{P3C,H,h}|_{P2,A} - t_{P4C,H,l}|_{P2,A}\,|, \pi_{P3C,H}, \pi_{P4C,H}) \quad \text{(C.126)}$$

C.8.18. Relative skew of the local-time reset for a set including processes P4 and P4C

Let $\pi_{P4+P4C,H}$ denote the bound on the relative skew of the local-time reset for a node set including all trustworthy or good recovering BIUs synchronizing with respect to process P4 or P4C. With respect to process P3, the Accept outputs in process P4 at the trustworthy BIUs and the Accept outputs in process P4C at the good recovering BIUs can be asserted during the same real time interval. This time range is determined by the time range during which the trustworthy RMU nodes executing process P3 send their ECHO messages. In the presence of asymmetric faulty RMU nodes, the good recovering BIUs executing process P4C may not able to synchronize any better than the duration of this time range.

$$\pi_{P4+P4C,H} = \max(\pi_{P4,H} , \pi_{P4C,H}) = \pi_{P4C,H} \quad \text{(C.127)}$$

C.8.19. Relative skew of the local-time reset for a set including all the synchronizing nodes

Let $\pi_{ALL,H}$ denote the upper bound on the relative skew of the local-time reset for all the trustworthy or good recovering nodes executing the synchronization protocol. The following relations allow us to reduce the number of relative skews that must be considered:

$$\pi_{P2+P3C,H} \geq \pi_{P2,H} \text{ and } \pi_{P2+P3C,H} \geq \pi_{P3C,H} \quad \text{(C.128)}$$

$$\pi_{P2+P3C,H} \geq \pi_{P2+P3,H} \quad \text{(C.129)}$$

$$\pi_{P2+P4C,H} \geq \pi_{P2,H} \text{ and } \pi_{P2+P4C,H} \geq \pi_{P4C,H} \quad \text{(C.130)}$$

$$\pi_{P2+P4C,H} \geq \pi_{P2+P4,H} \quad \text{(C.131)}$$

$$\pi_{P3+P4C,H} \geq \pi_{P3,H} \text{ and } \pi_{P3+P4C,H} \geq \pi_{P4C,H} \quad \text{(C.132)}$$

$$\pi_{P3+P4C,H} \geq \pi_{P3+P4,H} \quad \text{(C.133)}$$

$$\pi_{P3C+P4C,H} \geq \pi_{P3C,H} \text{ and } \pi_{P3C+P4C,H} \geq \pi_{P4C,H} \quad \text{(C.134)}$$

So:

$$\pi_{ALL,H} = \max(\pi_{P2+P3C,H}, \pi_{P2+P4C,H}, \pi_{P3+P4C,H}, \pi_{P3C+P4C,H}) \quad \text{(C.135)}$$

C.9. Relative local-time skews for source-receiver pairs

C.9.1. Duration of the synchronization protocol execution

From global perspective, the execution of the synchronization protocol ends when all the trustworthy and good recovering nodes have reset their local-time clocks. $t_{sync,l}$ and $t_{sync,h}$ denote the earliest and latest times, respectively, at which a trustworthy BIU node begins to execute the synchronization protocol. π_{P0} denotes the bound on the relative local-time skew for the trustworthy BIU nodes executing process P0.

$$\pi_{P0} = t_{sync,h} - t_{sync,l} \quad \text{(C.136)}$$

$t_{sync,P2,H,l}$ and $t_{sync,P2,H,h}$ denote the earliest and latest times, respectively, at which a trustworthy BIU node synchronizing with respect to process P2 can reset its local-time clock.

$$t_{sync,P2,H,l} = t_{sync,l} + 2r_{PP,l} + (B_{P0} + A_{P1} + B_{P1} + A_{P2} + H_{P2})/(1 + \rho_0) \quad \text{(C.137)}$$

$$t_{sync,P2,H,h} = t_{sync,h} + 2r_{PP,h} + (1 + \rho_0)(B_{P0} + A_{P1} + B_{P1} + A_{P2} + H_{P2}) \quad \text{(C.138)}$$

$t_{sync,P3,H,l}$ and $t_{sync,P3,H,h}$ denote the earliest and latest times, respectively, at which a trustworthy RMU node synchronizing with respect to process P3 can reset its local-time clock.

$$t_{sync,P3,H,l} = t_{sync,l} + 3r_{PP,l} + (B_{P0} + A_{P1} + B_{P1} + A_{P2} + B_{P2} + A_{P3} + H_{P3})/(1 + \rho_0) \quad \text{(C.139)}$$

$$t_{sync,P3,H,h} = t_{sync,h} + 3r_{PP,h} + (1 + \rho_0)(B_{P0} + A_{P1} + B_{P1} + A_{P2} + B_{P2} + A_{P3} + H_{P3}) \quad \text{(C.140)}$$

$t_{sync,P4,H,l}$ and $t_{sync,P4,H,h}$ denote the earliest and latest times, respectively, at which a trustworthy BIU node synchronizing with respect to process P4 can reset its local-time clock.

$$t_{sync,P4,H,l} = t_{sync,l} + 4r_{PP,l} + (B_{P0} + A_{P1} + B_{P1} + A_{P2} + B_{P2} + A_{P3} + B_{P3} + A_{P4} + H_{P4})/(1 + \rho_0) \quad \text{(C.141)}$$

$$t_{sync,P4,H,h} = t_{sync,h} + 4r_{PP,h} + (1 + \rho_0)(B_{P0} + A_{P1} + B_{P1} + A_{P2} + B_{P2} + A_{P3} + B_{P3} + A_{P4} + H_{P4}) \quad \text{(C.142)}$$

$t_{sync,P3C,H,l}$ and $t_{sync,P3C,H,h}$ denote the earliest and latest times, respectively, at which a good recovering RMU node synchronizing with respect to process P3C can reset its local-time clock.

$$t_{sync,P3C,H,l} = t_{sync,P3,H,l} \quad \text{(C.143)}$$

$$t_{sync,P3C,H,h} = t_{sync,P3,H,h} \quad \text{(C.144)}$$

$t_{sync,P4C,H,l}$ and $t_{sync,P4C,H,h}$ denote the earliest and latest times, respectively, at which a good recovering BIU node synchronizing with respect to process P4C can reset its local-time clock.

$$t_{sync,P4C,H,l} = t_{sync,P4,H,l} \quad \text{(C.145)}$$

$$t_{sync,P4C,H,h} = t_{sync,P4,H,h} \quad \text{(C.146)}$$

π_{ALL} denotes the bound on the relative local-time skew for all the nodes participating in the execution of the synchronization protocol. The calculation of π_{ALL} does not include the nodes executing the synchronization-capture processes. $\delta_{sync}|_{min}$ and $\delta_{sync}|_{max}$ denote lower and upper bounds, respectively, on the real-time duration of the execution of the synchronization protocol for the trustworthy nodes. $\delta_{sync}|_{min}$ is measured from the latest time at which a trustworthy node begins to execute the protocol to the earliest time at which a trustworthy node resets its local-time clock. We choose the following value for $\delta_{sync}|_{min}$:

$$\delta_{sync}|_{min} = -(\pi_{ALL} - \pi_{P0}) + \min[(t_{sync,P2,H,l} - t_{sync,h}), (t_{sync,P3,H,l} - t_{sync,h}), (t_{sync,P4,H,l} - t_{sync,h})] \quad (C.147)$$

$\delta_{sync}|_{max}$ is measured from the earliest time at which a trustworthy node begins to execute the protocol to the latest time at which a trustworthy node resets its local-time clock. We choose the following value for $\delta_{sync}|_{max}$:

$$\delta_{sync}|_{max} = (\pi_{ALL} - \pi_{P0}) + \max[(t_{sync,P2,H,h} - t_{sync,l}), (t_{sync,P3,H,h} - t_{sync,l}), (t_{sync,P4,H,h} - t_{sync,l})] \quad (C.148)$$

We define the following variables in order to simplify these expressions for $\delta_{sync}|_{min}$ and $\delta_{sync}|_{max}$.

$$\Delta_{sync,P2,H,l} = 2r_{PP,l} + (B_{P0} + A_{P1} + B_{P1} + A_{P2} + H_{P2})/(1 + \rho_0) \quad (C.149)$$

$$\Delta_{sync,P3,H,l} = 3r_{PP,l} + (B_{P0} + A_{P1} + B_{P1} + A_{P2} + B_{P2} + A_{P3} + H_{P3})/(1 + \rho_0) \quad (C.150)$$

$$\Delta_{sync,P4,H,l} = 4r_{PP,l} + (B_{P0} + A_{P1} + B_{P1} + A_{P2} + B_{P2} + A_{P3} + B_{P3} + A_{P4} + H_{P4})/(1 + \rho_0) \quad (C.151)$$

$$\Delta_{sync,P2,H,h} = 2r_{PP,h} + (1 + \rho_0)(B_{P0} + A_{P1} + B_{P1} + A_{P2} + H_{P2}) \quad (C.152)$$

$$\Delta_{sync,P3,H,h} = 3r_{PP,h} + (1 + \rho_0)(B_{P0} + A_{P1} + B_{P1} + A_{P2} + B_{P2} + A_{P3} + H_{P3}) \quad (C.153)$$

$$\Delta_{sync,P4,H,h} = 4r_{PP,h} + (1 + \rho_0)(B_{P0} + A_{P1} + B_{P1} + A_{P2} + B_{P2} + A_{P3} + B_{P3} + A_{P4} + H_{P4}) \quad (C.154)$$

Then:

$$\delta_{sync}|_{min} = -\pi_{ALL} + \min(\Delta_{sync,P2,H,l}, \Delta_{sync,P3,H,l}, \Delta_{sync,P4,H,l}) \quad (C.155)$$

$$\delta_{sync}|_{max} = \pi_{ALL} + \max(\Delta_{sync,P2,H,h}, \Delta_{sync,P3,H,h}, \Delta_{sync,P4,H,h}) \quad (C.156)$$

C.9.2. Bounds on the resynchronization period

Let $\delta_{SP}|_{min}$ and $\delta_{SP}|_{max}$ denote the values of $\delta_{sync}|_{min}$ and $\delta_{sync}|_{max}$, respectively, for the Synchronization Preservation protocol. T_{SP} denotes the scheduled local time to begin the execution of the Synchronization Preservation protocol. p_{min} denotes a lower bound on the real-time duration of a synchronization cycle. p_{min} is measured from the time of the synchronization reset in one cycle to the time of the synchronization reset in the next.

$$p_{min} = T_{SP}/(1 + \rho_0) + \delta_{SP}|_{min} \quad (C.157)$$

p_{max} denotes an upper bound for the real-time duration of a synchronization cycle. p_{max} is measured from the time of the synchronization reset in one cycle to the time of the synchronization reset in the next.

$$p_{max} = (1 + \rho_0)T_{SP} + \delta_{SP}|_{max} \quad (C.158)$$

P denotes the nominal resynchronization period for the analysis of relative skews. P is measured in units of local-clock ticks. We want a count of P local-clock ticks to be larger than the

maximum duration of a synchronization cycle measured in nominal ticks. This constraint is captured by the following expression:

$$P/(1 + \rho_0) \geq p_{max} \quad \text{(C.159)}$$

So:

$$P \geq (1 + \rho_0)p_{max} \quad \text{(C.160)}$$

We choose P to be the smallest integer that satisfies the previous inequality.

$$P = \lceil (1 + \rho_0)p_{max} \rceil \quad \text{(C.161)}$$

C.9.3. Relative skew between P2-synchronized BIUs and P3- or P3C-synchronized RMUs

Let $\pi_{P2\text{-}P3}$ denote the bound on the relative local-time skew during the synchronization cycle for trustworthy BIU nodes synchronized with respect to process P2 and trustworthy RMU nodes synchronized with respect to process P3.

$$\pi_{P2\text{-}P3} = \pi_{P2\text{-}P3,H} + [(1 + \rho_0) - 1/(1 + \rho_0)]P \quad \text{(C.162)}$$

$\pi_{P2\text{-}P3C}$ denotes the bound on the relative local-time skew during the synchronization cycle for trustworthy BIU nodes synchronized with respect to process P2 and good recovering RMU nodes synchronized with respect to process P3C. $\pi_{P2\text{-}P3}$ also applies here.

$$\pi_{P2\text{-}P3C} = \pi_{P2\text{-}P3} \quad \text{(C.163)}$$

C.9.4. Relative skew between P3-synchronized RMUs and P4- or P4C-synchronized BIUs

Let $\pi_{P3\text{-}P4}$ denote the bound on the relative local-time skew during the synchronization cycle for trustworthy RMU nodes synchronized with respect to process P3 and trustworthy BIU nodes synchronized with respect to process P4. Then:

$$\pi_{P3\text{-}P4} = \pi_{P3\text{-}P4,H} + [(1 + \rho_0) - 1/(1 + \rho_0)]P \quad \text{(C.164)}$$

$\pi_{P3\text{-}P4C}$ denotes the bound on the relative local-time skew during the synchronization cycle for trustworthy RMU nodes synchronized with respect to process P3 and good recovering BIU nodes synchronized with respect to process P4C. $\pi_{P3\text{-}P4}$ also applies here.

$$\pi_{P3\text{-}P4C} = \pi_{P3\text{-}P4} \quad \text{(C.165)}$$

C.9.5. Bound on the relative local-time skew for all the nodes executing the synchronization protocol

$\pi_{SP,ALL}$ denotes the value of π_{ALL} for Synchronization Preservation.

$$\pi_{SP,ALL} = \pi_{ALL,H} + [(1 + \rho_0) - 1/(1 + \rho_0)]P \quad \text{(C.166)}$$

C.9.6. Generic relative local-time skew between sources and receivers for synchronous communication

For synchronized operations, we would like to use a single value of the relative local-time skew between sources and receivers for all point-to-point communication. $\pi_{PP,SR}$ denotes the common bound on the relative local-time skew between sources and receivers for synchronized communication. From the preceding analysis, there are only two particular source-receiver cases that need to be considered to determine a common skew bound: the skew between P2-synchronized nodes and P3-synchronized nodes (i.e., $\pi_{P2\text{-}P3}$), and the skew between P3-synchronized nodes and P4-synchronized nodes (i.e., $\pi_{P3\text{-}P4}$). We choose $\pi_{PP,SR}$ to be the largest of the two.

$$\pi_{PP,SR} = \max(\pi_{P2\text{-}P3,}\, \pi_{P3\text{-}P4}) \tag{C.167}$$

C.10. Specifying the Computation Process and Send Process delays

A goal of this ROBUS version is to achieve nearly the same tightness for the relative local-time skew when executing the Synchronization Preservation, Initial Synchronization, and Synchronization Capture protocols.

The Synchronization Preservation and Initial Synchronization protocols can be decomposed into two major phases: agreement generation and agreement propagation. The agreement generation phase includes the first two stages of the protocol from the Send Process in P0 to the Computation Process in P2. In this phase, the relative skew goes from a bounded initial value denoted by π_{P0} to a relative skew of the Accept outputs denoted by $\pi_{P2,A}$, which is independent of π_{P0} but dependent on the process delays. The agreement propagation phase includes the last two stages of the protocol from the Send Process in P2 to the Computation Process in P4, including the Computation Processes in P3C and P4C for the Synchronization Capture protocol. The synchronization-reset delays are applied with respect to the Accept outputs in processes P2, P3, P3C, P4, and P4C. The process delays for this second phase of the protocol are important determinants of the final relative local-time skew.

The approach taken to determine the process delays for the synchronization protocols in this version of the ROBUS is as follows. Since we expect the value of π_{P0} to be different for the Synchronization Preservation and the Initial Synchronization protocols, we specify the Send Process delay for process P0 (i.e., B_{P0}) independently for each protocol according to the particular timing requirements of the protocol. To ensure that all the versions of the protocol achieve approximately the same relative skew, we compute one set of Computation Process and Send Process delays for the synchronization processes from P1 on. These delays must be used by all the synchronization protocols.

An additional consideration is the constraint on the minimum data-introduction interval (DII) for the send port of the Communication Module, Λ_{Comm}. This constraint applies to the BIUs and the RMUs, and is satisfied by adding functional requirements to the Send Processes at the BIUs and the RMUs. The details are described next.

C.10.1. Computation Process delays

The Computation Process delay is decomposed into two parts: the reception delay in the Receive Process and the computation delay in the Accept Process. For the case of the Synchronization Preservation protocol, the reception delay is the delay allocated to ensure that all valid messages are received before the computation begins. This delay is similar to the deskewing window applied for the synchronous point-to-point communication as discussed in Appendix B. A fundamental difference between the reception delay for the synchronization protocols and the deskewing window for the synchronous protocols is that in the synchronization protocols the relative time spacing between received messages is preserved when forwarding the messages to the computation, while in the synchronous protocols the messages are accumulated and forwarded at the same time.

To specify the reception delay, we consider the timing of reception in the Synchronization Preservation protocol. The timing of reception in the Initial Synchronization protocol is not considered because in that protocol the uncertainty in the time of reception can be extremely large, especially for processes P1 and P2, which would result in very large delays for the protocol. Having a quick execution is very important for the Synchronization Preservation protocol since the duration of the protocol determines how much time is available to execute the synchronous protocols for a given resynchronization period.

Let $A_{IS,P1}$, $A_{IS,P2}$, $A_{IS,P3}$, and $A_{IS,P4}$ denote the Computation Process delays for processes P1, P2, P3, and P4 of the Initial Synchronization protocol, respectively. $A_{SP,P1}$, $A_{SP,P2}$, $A_{SP,P3}$, and $A_{SP,P4}$ denote the Computation Process delays for processes P1, P2, P3, and P4 of the Synchronization Preservation protocol, respectively. $A_{SC,P3C}$ and $A_{SC,P4C}$ denote the Computation Process delays for processes P3C and P4C of the Synchronization Capture protocol, respectively. All the synchronization protocols have the same Computation Process delays.

$$A_{P1} = A_{IS,P1} = A_{SP,P1} \tag{C.168}$$

$$A_{P2} = A_{IS,P2} = A_{SP,P2} \tag{C.169}$$

$$A_{P3} = A_{IS,P3} = A_{SP,P3} = A_{SP,P3C} \tag{C.170}$$

$$A_{P4} = A_{IS,P4} = A_{SP,P4} = A_{SP,P4C} \tag{C.171}$$

For process P1 of the Synchronization Preservation protocol, the expected time range of reception is as follows (This interval includes all the clock edges at which valid messages can arrive.):

$$[T_{SP,P1,RCV,E} - \Delta_{PP,RCV}|_{abs\text{-}max}, T_{SP,P1,RCV,E} + \Delta_{PP,RCV}|_{abs\text{-}max}] \tag{C.172}$$

$W_{SP,P1}$ denotes the reception delay applied in process P1. For the Synchronization Preservation protocol, this delay must be large enough to ensure that the Accept Process receives the messages after the clock edges during which valid messages are expected to arrive.

$$W_{SP,P1} = 2\Delta_{PP,RCV}|_{abs\text{-}max} + 1 \tag{C.173}$$

$W_{SP,P2}$, $W_{SP,P3}$, and $W_{SP,P4}$ are similarly defined. Let $\Delta_{SP,P2,RCV}|_{max}$, $\Delta_{SP,P3,RCV}|_{max}$, and $\Delta_{SP,P4,RCV}|_{max}$ denote the maximum valid local time error for the time of reception of synchronization messages

in processes P2, P3, and P4 of the Synchronization Preservation protocol. These variables correspond to $\Delta_{P2,RCV}|_{max}$, $\Delta_{P3,RCV}|_{max}$, and $\Delta_{P4,RCV}|_{max}$ evaluated for the case of the Synchronization Preservation protocol. Then:

$$W_{SP,P2} = 2\Delta_{SP,P2,RCV}|_{max} + 1 \quad \text{(C.174)}$$

$$W_{SP,P3} = 2\Delta_{SP,P3,RCV}|_{max} + 1 \quad \text{(C.175)}$$

$$W_{SP,P4} = 2\Delta_{SP,P4,RCV}|_{max} + 1 \quad \text{(C.176)}$$

Notice that for process P1 the expected time of reception is exactly $\Delta_{SP,P1,RCV}|_{max}$ ticks from the left edge of the reception interval in that process. Similar observations apply to processes P2 through P4. The computation delay is measured from the time the message to be selected is presented to the Accept Process until the Accept output is asserted. $C_{SP,P1}$, $C_{SP,P2}$, $C_{SP,P3}$, and $C_{SP,P4}$ denote the Accept Process delays for processes P1, P2, P3, and P4, respectively, of the Synchronization Preservation protocol. These delays also apply to the Initial Synchronization and Synchronization Capture protocols. Then:

$$A_{P1} = W_{SP,P1} + C_{SP,P1} \quad \text{(C.177)}$$

$$A_{P2} = W_{SP,P2} + C_{SP,P2} \quad \text{(C.178)}$$

$$A_{P3} = W_{SP,P3} + C_{SP,P3} \quad \text{(C.179)}$$

$$A_{P4} = W_{SP,P4} + C_{SP,P4} \quad \text{(C.180)}$$

C.10.2. Send Process delays

The Send Process delays must be set to ensure proper inter-process communication. The Send Process delay for process P0 does not need to be the same for Initial Synchronization and Synchronization Preservation. The specification of that value for each protocol is presented below. For all the other Send Processes, we specify the delays based on two factors. First, we would like to specify the process delays based on the execution of the Synchronization Preservation protocol. The timing of execution of the Initial Synchronization protocol is not preferred because in that protocol the uncertainty in the time of reception can be extremely large, which would result in extremely large process delays for the protocol. The second factor when specifying the Send Process delays is the need to satisfy the minimum data-introduction-interval constraint for the send port of the Communication Module, Λ_{Comm}, which must be satisfied at the BIUs and the RMUs.

For the execution of the Synchronization Preservation protocol, the main concern in specifying the Send Process delays is ensuring proper coordination between the send and receive operations. In particular, the specification of the send delay must take into consideration the expected reception delay, the minimum delays in opening the input windows, and the size of the input windows. This is not a consideration in the Initial Synchronization protocol since in that case all the Computation Processes are enabled at the beginning of the execution of the protocol.

For the Synchronization Preservation protocol, we must ensure that the time separation

between the sending of INIT and ECHO messages satisfies the Λ_{Comm} constraint. The preferred method to satisfy this constraint in the Synchronization Preservation protocol is to increase the Send Process delays for the INIT messages in processes P0 and P1 and/or the ECHO messages in processes P2 and P3 until sufficient separation between them is ensured.

For the Initial Synchronization protocol, the problem is more complicated. Because the initial relative local-time skew can be much larger than the Computation Process and Send Process delays, there is no way to meet the Λ_{Comm} constraint by simply changing the process delays while still achieving the other design goals. The preferred solution for this case is to add functionality to the Send Processes at the BIUs and the RMUs to force a minimum separation between INIT and ECHO messages. However, the buffering of synchronization messages for a bounded but unspecified amount of time at a Send Process is an undesired solution because it would result in an increase in the bound on the relative local-time skew achieved by the protocol. Instead, the solution is based on the observation that for the Initial Synchronization protocol, once the Computation Process of a node has performed the computation that triggers the sending of an ECHO message (i.e., Accept(INIT) in process P2 at the BIUs, and Accept(ECHO) in process P3 at the RMUs), there is no need for the node to send an INIT message. To understand this, notice that the synchronization protocol achieves synchronization in process P2, and this is then propagated to processes P3 and P4 using ECHO messages. For the Initial Synchronization protocol, RMUs and BIUs reset their local times with respect to the Accept(ECHO) outputs in processes P3 and P4, respectively. Therefore, the fact that an ECHO message is going to be sent means that whatever critical timing information was going to be provided by processes P0 and P1, it has already been received. Therefore, the INIT messages are redundant from that point on. So, for Initial Synchronization, to meet the minimum data-introduction-interval constraint, the Send Process must have the following features:

- The sending of an INIT message must be blocked if the message has not been sent by the time the Accept output that triggers the sending of an ECHO message is asserted.

- The send delay for ECHO messages must be larger than or equal to Λ_{Comm} - 1.

The first functional requirement removes redundant INIT messages. The second requirement ensures that, if an INIT message is sent at or before the tick at which the Accept output that triggers the sending of an ECHO message is asserted, then the ECHO message will be sent at least Λ_{Comm} ticks after the INIT message.

Let $B_{IS,P0}$, $B_{IS,P1}$, $B_{IS,P2}$, and $B_{IS,P3}$ denote the Send Process delays for processes P0, P1, P2, and P3 of the Initial Synchronization protocol, respectively. $B_{SP,P0}$, $B_{SP,P1}$, $B_{SP,P2}$, and $B_{SP,P3}$ denote the Computation Process delay for processes P0, P1, P2, and P3 of the Synchronization Preservation protocol, respectively. For processes P1, P2, and P3:

$$B_{P1} = B_{IS,P1} = B_{SP,P1} \tag{C.181}$$

$$B_{P2} = B_{IS,P2} = B_{SP,P2} \tag{C.182}$$

$$B_{P3} = B_{IS,P3} = B_{SP,P3} \tag{C.183}$$

$B_{IS,P0}$ and $B_{SP,P0}$ are specified separately for Initial Synchronization and Synchronization Preservation.

C.10.2.1. Send delay for process P0

C.10.2.1.1. Synchronization Preservation

The Synchronization Preservation protocol is a time-triggered, event-driven protocol. The communication between processes P0 and P1 follows a time-triggered pattern similar to the point-to-point communication of the synchronous protocols. After that operation, the rest of the Synchronization Preservation protocol proceeds driven by communication and processing events. T_{SP} denotes the local-time trigger for the execution of the Synchronization Preservation protocol. $B_{SP,P0}$ denotes the send delay for process P0 of the Synchronization Preservation protocol. $B_{SP,P0}|_{min}$ denotes the minimum send delay for process P0. $B_{SP,P0}|_{min}$ is assumed to be the time needed to prepare the message for transmission. $B_{SP,P0} - B_{SP,P0}|_{min}$ is additional delay added to align the send and receive operations. $\Delta_{SP,P1,RCVWND}$ denotes the delay from the communication reference time to the opening of the receive window in process P1. $\Delta_{SP,P1,RCVWND}|_{min}$ is the minimum value of $\Delta_{SP,P1,RCVWND}$. R_{PP} denotes the expected point-to-point reception delay. $W_{SP,P1}$ is the size of the reception window. $W_{SP,P1,pre}$ is the pre-expectation window (i.e., the size of the section of the reception window before the expected time of reception). Considering the analysis in Appendix B, $W_{SP,P1,pre}$ corresponds to $W_{Deskew,pre}$. So:

$$W_{SP,P1,pre} = \Delta_{PP,RCV}|_{abs\text{-}max.} \quad \text{(C.184)}$$

$T_{SP,P0,SND}$ denotes the send time for process P0. $T_{SP,P0,SND}$ corresponds to T_{P0} in the general analysis of the clock synchronization protocols. $T_{SP,P1,RCV,E}$ denotes the expected time of reception for process P1. $T_{SP,P0\text{-}P1,REF}$ denotes the reference time for the transmission between P0 and P1.

$$T_{SP,P0\text{-}P1,REF} = T_{SP} \quad \text{(C.185)}$$

Two cases must be considered.

Case 1: $B_{SP,P0}|_{min} + R_{PP} \geq \Delta_{SP,P1,RCVWND}|_{min} + W_{SP,P1,pre}$

For this case:

$$B_{SP,P0} = B_{SP,P0}|_{min} \quad \text{(C.186)}$$

$$\Delta_{SP,P1,RCVWND} = B_{SP,P0}|_{min} + R_{PP} - W_{SP,P1,pre} \quad \text{(C.187)}$$

So:

$$T_{SP,P0,SND} = T_{SP,P0\text{-}P1,REF} + B_{SP,P0} = T_{SP} + B_{SP,P0}|_{min} \quad \text{(C.188)}$$

And:

$$T_{SP,P1,RCV,E} = T_{SP,P0,SND} + R_{PP} = T_{SP} + B_{SP,P0}|_{min} + R_{PP} \quad \text{(C.189)}$$

Case 2: $B_{SP,P0}|_{min} + R_{PP} < \Delta_{SP,P1,RCVWND}|_{min} + W_{SP,P1,pre}$

For this case:

$$B_{SP,P0} = \Delta_{SP,P1,RCVWND}|_{min} + W_{SP,P1,pre} - R_{PP} \quad \text{(C.190)}$$

$$\Delta_{SP,P1,RCVWND} = \Delta_{SP,P1,RCVWND}|_{min} \quad \text{(C.191)}$$

So:

$$T_{SP,P0,SND} = T_{SP,P0\text{-}P1,REF} + B_{SP,P0} = T_{SP} + \Delta_{SP,P1,RCVWND}|_{min} + W_{SP,P1,pre} - R_{PP} \quad \text{(C.192)}$$

And:

$$T_{SP,P1,RCV,E} = T_{SP,P0,SND} + R_{PP} = T_{SP} + \Delta_{SP,P1,RCVWND}|_{min} + W_{SP,P1,pre} \quad \text{(C.193)}$$

C.10.2.1.2. Initial Synchronization

Let π_{IS} denote the bound on the relative local-time skew considering BIUs and RMUs during the execution of the Initial Synchronization protocol, measured in nominal clock ticks. T_{IS} denotes the local time triggering the execution of the Initial Synchronization protocol. The timing of the first-stage communication can be analyzed similarly to the point-to-point communication for synchronous protocols.

$T_{IS,P0\text{-}P1,REF}$ denotes the reference time for the communication between processes P0 and P1. $T_{IS,P0,SND}$ denotes the local time at which process P0 sends the message. $T_{IS,P0,SND}$ corresponds to T_{P0} in the general analysis of the clock synchronization protocols. $T_{IS,P1,RCV,E}$ denotes the expected time of reception in process P1. $B_{IS,P0}$ denotes the Send Process delay for process P0. $B_{IS,P0}|_{min}$ denotes the minimum send delay for process P0. $\Delta_{IS,P1,RCVWND}$ denotes the delay from the communication reference time to the opening of the receive window in process P1. $W_{IS,P1}$ denotes the size of the reception window in process P1. $W_{IS,P1,pre}$ denotes the pre-expectation window in process P1 (i.e., the size of the section of the reception window before the expected time of reception). We use T_{IS} as the reference time for the communication between processes P0 and P1.

$$T_{IS,P0\text{-}P1,REF} = T_{IS} \quad \text{(C.194)}$$

We use the analysis for point-to-point communication in Appendix B to determine $W_{IS,P1,pre}$. To determine $W_{IS,P1}$, we need the maximum error in the expected time of reception for the Initial Synchronization protocol messages, $\Delta_{IS,PP,RCV}|_{abs\text{-}max}$.

$$\Delta_{IS,PP,RCV}|_{abs\text{-}max} = \lfloor (1 + \rho_0)(\pi_{IS} + \max(\mu_{PP,l}\ ,\ \mu_{PP,h})) \rfloor \quad \text{(C.195)}$$

$\mu_{PP,l}$ and $\mu_{PP,h}$ are given in the Appendix B. So, for the reception window:

$$W_{IS,P1} = 2\Delta_{IS,PP,RCV}|_{abs\text{-}max} + 1 \quad \text{(C.196)}$$

$$W_{IS,P1,pre} = \Delta_{IS,PP,RCV}|_{abs\text{-}max} \quad \text{(C.197)}$$

$B_{IS,P0}|_{min}$ is assumed to be the time needed to prepare the message for transmission.

We expect the upper bound on the relative local-time skew during the execution of the first stage of the Initial Synchronization protocol to be much larger than any minimum timing constraints associated with the process of communication. Based on this, we assume that the following condition holds for the communication between processes P0 and P1.

$$B_{IS,P0}|_{min} + R_{PP} < \Delta_{IS,P1,RCVWND}|_{min} + W_{IS,P1,pre} \quad \text{(C.198)}$$

For this case:

$$B_{IS,P0} = \Delta_{IS,P1,RCVWND}|_{min} + W_{IS,P1,pre} - R_{PP} \quad \text{(C.199)}$$

$$\Delta_{IS,P1,RCVWND} = \Delta_{IS,P1,RCVWND}|_{min} \quad \text{(C.200)}$$

So:

$$T_{IS,P0,SND} = T_{IS,P0\text{-}P1,REF} + B_{IS,P0} = T_{IS} + \Delta_{IS,P1,RCVWND}|_{min} + W_{IS,P1,pre} - R_{PP} \quad \text{(C.201)}$$

And:

$$T_{IS,P1,RCV,E} = T_{IS,P0,SND} + R_{PP} = T_{IS} + \Delta_{IS,P1,RCVWND}|_{min} + W_{IS,P1,pre} \quad \text{(C.202)}$$

C.10.2.2. Send delay for process P1

B_{P1} is specified based on timing considerations for Synchronization Preservation. $B_{P1}|_{min}$ is determined by the implementation. Process P1 sends INIT to process P2. However, the reference event used to coordinate the communication between processes P1 and P2 is the trigger time for the transmission of the message in process P0. Let $T_{SP,P0\text{-}P2,REF}$ denote this reference.

$$T_{SP,P0\text{-}P2,REF} = T_{SP} + B_{SP,P0} \quad \text{(C.203)}$$

$R_{SP,P0\text{-}P2}$ denotes the expected reception delay for process P2 of the Synchronization Preservation protocol. $R_{SP,P0\text{-}P2}$ is measured from the send time in process P0 to the expected time of reception in process P2. $\Delta_{SP,P2,RCVWND}$ denotes the delay from the reference time to the opening of the input window in process P2. $\Delta_{SP,P2,RCVWND}|_{min}$ denotes the minimum value, which is determined by the implementation. For proper communication, the following relation must be satisfied:

$$R_{SP,P0\text{-}P2} = \Delta_{SP,P2,RCVWND} + \Delta_{SP,P2,RCV}|_{max} \quad \text{(C.204)}$$

Here, both $R_{SP,P0\text{-}P2}$ and $\Delta_{SP,P2,RCV}|_{max}$ are functions of B_{P1}, and $\Delta_{SP,P2,RCVWND}$ can be made larger than $\Delta_{SP,P2,RCVWND}|_{min}$. Solving this equation for B_{P1} is not trivial. However, note that $R_{SP,P0\text{-}P2}$ varies one-to-one with respect to B_{P1}, while $\Delta_{SP,P2,RCV}|_{max}$ changes by approximately $2\rho_0 B_{P1}$ for each unit step in B_{P1}. This observation allows us to use the following algorithm to determine B_{P1}. The notation $R_{SP,P0\text{-}P2}(B_{P1})$ and $\Delta_{SP,P2,RCV}|_{max}(B_{P1})$ highlights the dependence of $R_{SP,P0\text{-}P2}$ and $\Delta_{SP,P2,RCV}|_{max}$ on B_{P1}.

1. $B_{P1} = B_{P1}|_{min}$
2. while $[R_{SP,P0\text{-}P2}(B_{P1}) < \Delta_{SP,P2,RCV}|_{max}(B_{P1}) + \Delta_{SP,P2,RCVWND}|_{min}]$

3. $\{B_{P1} = B_{P1} + 1\}$

4. Results:
5. B_{P1}
6. $R_{SP,P0\text{-}P2} = R_{SP,P0\text{-}P2}(B_{P1})$
7. $\Delta_{SP,P2,RCV}|_{max} = \Delta_{SP,P2,RCV}|_{max}(B_{P1})$
8. $\Delta_{SP,P2,RCVWND} = R_{SP,P0\text{-}P2} - \Delta_{SP,P2,RCV}|_{max}$

C.10.2.3. Send delay for process P2

B_{P2} is specified based on timing considerations for Synchronization Preservation and the minimum data-introduction-interval constraint of the Communication Module. $B_{P2}|_{min}$ is determined by the implementation. $T_{SP,P1\text{-}P3,REF}$ denotes the reference time for the communication message propagation from P1 to P3. The event used to coordinate this communication is the time of the Accept output in process P1, denoted by $T_{SP,P1,A}$ for the Synchronization Preservation protocol.

$$T_{SP,P1\text{-}P3,REF} = T_{SP,P1,A} \tag{C.205}$$

$R_{SP,P1\text{-}P3}$ denotes the expected reception delay for process P3 of the Synchronization Preservation protocol. $R_{SP,P1\text{-}P3}$ is measured from the time of Accept output in process P1 to the expected time of reception in process P3. $\Delta_{SP,P3,RCVWND}$ denotes the delay from the reference time to the opening of the input window in process P3. $\Delta_{SP,P3,RCVWND}|_{min}$ denotes the minimum value, which is determined by the implementation. For proper communication, the following relation must be satisfied:

$$R_{SP,P1\text{-}P3} = \Delta_{SP,P3,RCVWND} + \Delta_{SP,P3,RCV}|_{max} \tag{C.206}$$

As for the case of B_{P1}, solving this equation for B_{P2} is non-trivial. Therefore, we use here the same algorithm used to solve for B_{P1}. An additional constraint is that B_{P2} must be larger than or equal to $\Lambda_{Comm} - 1$.

1. $B_{P2} = B_{P2}|_{min}$
2. if $(B_{P2} < \Lambda_{Comm} - 1)$, then $B_{P2} = \Lambda_{Comm} - 1$.
2. while $[R_{SP,P1\text{-}P3}(B_{P2}) < \Delta_{SP,P3,RCV}|_{max}(B_{P2}) + \Delta_{SP,P3,RCVWND}|_{min}]$
3. $\{B_{P2} = B_{P2} + 1\}$

4. Results:
5. B_{P2}
6. $R_{SP,P1\text{-}P3} = R_{SP,P1\text{-}P3}(B_{P2})$
7. $\Delta_{SP,P3,RCV}|_{max} = \Delta_{SP,P3,RCV}|_{max}(B_{P2})$
8. $\Delta_{SP,P3,RCVWND} = R_{SP,P1\text{-}P3} - \Delta_{SP,P3,RCV}|_{max}$

C.10.2.4. Send delay for process P3

B_{P3} is specified based on timing considerations for Synchronization Preservation and the minimum data-introduction-interval constraint of the Communication Module. $B_{P3}|_{min}$ is determined by the implementation. $T_{SP,P2\text{-}P4,REF}$ denotes the reference time for the communication

message propagation from P2 to P4. The event used to coordinate this communication is the time of the Accept output in process P1, denoted by $T_{SP,P2,A}$ for the Synchronization Preservation protocol.

$$T_{SP,P2\text{-}P4,REF} = T_{SP,P2,A} \quad \text{(C.207)}$$

$R_{SP,P2\text{-}P4}$ denotes the expected reception delay for process P4 of the Synchronization Preservation protocol. $R_{SP,P2\text{-}P4}$ is measured from the time of Accept in process P2 to the expected time of reception in process P4. Let $\Delta_{SP,P4,RCVWND}$ denote the delay from the reference time to the opening of the input window in process P4. $\Delta_{SP,P4,RCVWND}|_{min}$ denotes the minimum value, which is determined by the implementation. For proper communication, the following relation must be satisfied:

$$R_{SP,P2\text{-}P4} = \Delta_{SP,P4,RCVWND} + \Delta_{SP,P4,RCV}|_{max} \quad \text{(C.208)}$$

As for the case of B_{P2}, solving this equation for B_{P3} is non-trivial. Therefore, we use here the same algorithm used to solve for B_{P2}. An additional constraint is that B_{P3} must be larger than or equal to Λ_{Comm} - 1.

1. $B_{P3} = B_{P3}|_{min}$
2. if ($B_{P3} < \Lambda_{Comm} - 1$), then $B_{P3} = \Lambda_{Comm} - 1$.
2. while $[R_{SP,P2\text{-}P4}(B_{P3}) < \Delta_{SP,P4,RCV}|_{max}(B_{P3}) + \Delta_{SP,P4,RCVWND}|_{min}]$
3. $\{B_{P3} = B_{P3} + 1\}$

4. Results:
5. B_{P3}
6. $R_{SP,P2\text{-}P4} = R_{SP,P2\text{-}P4}(B_{P3})$
7. $\Delta_{SP,P4,RCV}|_{max} = \Delta_{SP,P4,RCV}|_{max}(B_{P3})$
8. $\Delta_{SP,P4,RCVWND} = R_{SP,P2\text{-}P4} - \Delta_{SP,P4,RCV}|_{max}$

C.11. Miscellaneous considerations

C.11.1. Frame Synchronization

The Frame Synchronization protocol is presented in Section 7 of this document. The protocol is executed by recovering nodes in the Synchronization Acquisition mode. The end of the Frame Synchronization protocol triggers the execution of the P3C or P4C synchronization-capture processes. An assumption for the protocol is the existence of a single valid clique in Preservation mode. The protocol monitors the ECHO messages from the trusted nodes identified during Local Diagnosis Acquisition. Achieving frame synchronization is equivalent to finding the time gap between consecutive executions of the Synchronization Preservation protocol. The Frame Synchronization protocol consists of searching for a time interval during which the clique is not sending ECHO messages. Finding such interval indicates that the clique is in between computations of clock adjustments, and thus it is an appropriate time to start the execution of the Synchronization Capture protocol. The Frame Synchronization protocol can achieve synchronization even if, for the node executing the protocol, it is not true that a majority of the

eligible sources of the opposite kind is trustworthy. The time interval measured by the gap timer, called the **frame synchronization gap**, corresponds to the maximum observed relative skew between received ECHO messages from trustworthy nodes. The analysis presented here applies to BIUs and RMUs.

Let $\Delta_{FS,GAP}$ denote the duration of the frame synchronization gap, measured in local clock ticks. $\Delta_{FS,GAP}|_{RMU}$ and $\Delta_{FS,GAP}|_{BIU}$ correspond to $\Delta_{FS,GAP}$ for RMUs and BIUs, respectively.

$$\Delta_{FS,GAP}|_{RMU} = \Pi_{SP,P3C,RCV} \tag{C.209}$$

$$\Delta_{FS,GAP}|_{BIU} = \Pi_{SP,P4C,RCV} \tag{C.210}$$

We choose $\Delta_{FS,GAP}|_{max}$ to be the largest value of $\Delta_{FS,GAP}$.

$$\Delta_{FS,GAP}|_{max} \geq \max(\Pi_{SP,P3C,RCV}, \Pi_{SP,P4C,RCV}) \tag{C.211}$$

We are interested in the worst-case duration of the Frame Synchronization protocol. Δ_{FS} denotes the actual duration of the execution of the Frame Synchronization protocol measured in local clock ticks. To determine the maximum duration of the Frame Synchronization protocol, we need to consider the possible patterns of interruption of the interval timer.

N denotes the total number of BIU nodes, and M denotes the total number of RMUs nodes. Let ω denote the number of eligible sources of the opposite kind.

$$\omega|_{max} = \max(N, M) \tag{C.212}$$

Δ_{FS} denotes the actual duration of the execution of the Frame Synchronization protocol, measured in local-clock ticks. An assumption for the Frame Synchronization protocol is that, during its execution, it will encounter at most one execution of the Synchronization Preservation protocol. Therefore, a source is allowed to interrupt the interval timer at most once during the execution of the protocol. In the worst-case, interruptions from eligible sources can consume up to $\omega|_{max} * \Delta_{FS,GAP}|_{max}$ local ticks in failed attempts to find a quiet frame synchronization gap (i.e., an interval with no gap timer interruptions). Adding an additional $\Delta_{FS,GAP}$ for the last interval, for which interruptions would not be allowed, then:

$$\Delta_{FS}|_{max} = (\omega|_{max} + 1) * \Delta_{FS,GAP}|_{max} \tag{C.213}$$

δ_{FS} denotes the worst-case duration of the Frame Synchronization protocol measured in nominal clock ticks.

$$\delta_{FS}|_{max} = (1 + \rho_0)\Delta_{FS}|_{max} \tag{C.214}$$

The assumption that, during its execution, the Frame Synchronization protocol will encounter at most one execution of the Synchronization Preservation protocol imposes the following constraint on the minimum duration of the resynchronization period p_{min}.

$$p_{min} \geq \delta_{FS}|_{max} \tag{C.215}$$

C.11.2. Executing Synchronization Preservation after Synchonization Acquisition

Recovering BIUs synchronize to an existing clique by executing the Synchronization Capture protocol and synchronizing with respect to process P4C. After synchronizing, the recovering BIUs behave synchronously just like the existing BIU members of the clique. In the first execution of the Synchronization Preservation protocol, the recovering BIUs synchronize with respect to process P2. There are two important differences between the existing trustworthy BIU members of a clique and good recovering BIUs during this execution of the Synchronization Preservation protocol.

The first difference is that good recovering BIUs do not transmit synchronization messages. Even if they transmitted messages, the existing members of the clique would not include those messages in the computation of the protocol. Therefore, for the existing trustworthy clique members, the result of the synchronization protocol does not depend on the performance of recovering BIUs.

The second (and more important) difference is that the good recovering BIUs are not necessarily synchronized to the existing trustworthy BIU clique members as tightly as the existing trustworthy BIU clique members are synchronized to each other. This difference is significant in the execution of process P2 and in the definitions of the expected reception delay $R_{SP,P0\text{-}P2}$ and the worst-case local-time difference between the actual time of reception and the expect time of reception $\Delta_{SP,P2,RCV}|_{max}$. As presented previously, the definition of these parameters uses the relative local-time skew of the transmitting BIUs in process P0 (i.e., π_{P0}). Because good recovering BIUs are not included in the definition of π_{P0}, their effective reception delay and its the worst-case error can be different than for the trustworthy BIU clique members. To correct this problem, the relative local-time skew used to compute $R_{SP,P0\text{-}P2}$ and $\Delta_{SP,P2,RCV}|_{max}$ must include the good recovering BIUs.

$\pi_{SP,P0}|_{P2,RCV}$ denotes the value of π_{P0} used to compute $R_{SP,P0\text{-}P2}$ and $\Delta_{SP,P2,RCV}|_{max}$ for process P2 of the Synchronization Preservation protocol.

$$\pi_{SP,P0}|_{P2,RCV} = \pi_{P2+P4C,H} + [(1 + \rho_0) - 1/(1 + \rho_0)]P \qquad \text{(C.216)}$$

$\pi_{SP,P0}$ denotes the value of π_{P0} for the Synchronization Preservation protocol. Except for the case above, $\pi_{SP,P0}$ is:

$$\pi_{SP,P0} = \pi_{P2,H} + [(1 + \rho_0) - 1/(1 + \rho_0)]P \qquad \text{(C.217)}$$

C.11.3. Time service accuracy for the Synchronization Preservation protocol

The PEs receive periodic time updates from the BIUs in the form of INIT messages. These messages are triggered by the output of the Accept(INIT) functions in process P2 of the Synchronization Preservation protocol. The accuracy of the time service is defined here as the maximum error in the expect period between Accept(INIT) outputs in consecutive executions of the Synchronization Preservation protocol.

Consider two consecutive executions of the Synchronization Preservation protocol, denoted by SP1 and SP2. $\pi_{P2,A}$ denotes the bound on the real-time relative skew of the Accept outputs in

process P2 at the trustworthy BIU nodes. $\pi_{P2,A}$ applies to SP1 and SP2. Let $t_{P2,A,l}|_{SP1}$ and $t_{P2,A,h}|_{SP1}$ denote the bounds on the earliest and latest real times, respectively, at which the trustworthy BIU nodes synchronizing with respect to process P2 of SP1 assert the output of their Accept(INIT) functions. Thus:

$$\pi_{P2,A} = t_{P2,A,h}|_{SP1} - t_{P2,A,l}|_{SP1} \tag{C.218}$$

Let $t_{P2,A,l}|_{SP2}$ and $t_{P2,A,h}|_{SP2}$ denote the bounds on the earliest and latest real times, respectively, at which the Accept outputs are asserted in process P2 at the trustworthy BIUs for SP2. The relations between $t_{P2,A,h}|_{SP1}$ and $t_{P2,A,l}|_{SP1}$, and $t_{P2,A,h}|_{SP2} - t_{P2,A,l}|_{SP2}$ are constrained by the drift rate of the local-time clocks and the validity interval for the Accept outputs in process P2 of SP2.

$$t_{P2,A,l}|_{SP2} = t_{P2,A,l}|_{SP1} + 2r_{PP,l} + (H_{P2} + T_{SP} + B_{P0} + A_{P1} + B_{P1} + A_{P2})/(1 + \rho_0) \tag{C.219}$$

$$t_{P2,A,h}|_{SP2} = t_{P2,A,h}|_{SP1} + 2r_{PP,h} + (1 + \rho_0)(H_{P2} + T_{SP} + B_{P0} + A_{P1} + B_{P1} + A_{P2}) \tag{C.220}$$

$P_{SVC}|_{min}$ and $P_{SVC}|_{max}$ denote the minimum and maximum intervals, respectively, between time updates for the time-reference service, measured in units of nominal clock ticks.

$$P_{SVC}|_{min} = t_{P2,A,l}|_{SP2} - t_{P2,A,h}|_{SP1}$$

$$= 2r_{PP,l} + (H_{P2} + T_{SP} + B_{P0} + A_{P1} + B_{P1} + A_{P2})/(1 + \rho_0) - \pi_{P2,A} \tag{C.221}$$

$$P_{SVC}|_{max} = t_{P2,A,h}|_{SP2} - t_{P2,A,l}|_{SP1}$$

$$= 2r_{PP,h} + (1 + \rho_0)(H_{P2} + T_{SP} + B_{P0} + A_{P1} + B_{P1} + A_{P2}) + \pi_{P2,A} \tag{C.222}$$

P_{SVC} denotes the expected period between time updates for the time-reference service, measured in units of nominal clock ticks.

$$P_{SVC} = (P_{SVC}|_{min} + P_{SVC}|_{max})/2 \tag{C.223}$$

Let α denote the accuracy of P_{SVC}.

$$\alpha = (P_{SVC}|_{max} - P_{SVC}|_{min})/2$$

$$= \pi_{P2,A} + e_{PP} + (1/2)[(1 + \rho_0) - 1/(1 + \rho_0)](H_{P2} + T_{SP} + B_{P0} + A_{P1} + B_{P1} + A_{P2}) \tag{C.224}$$

Substituting for $\pi_{P2,A}$:

$$\alpha = 3e_{PP} + [(1 + \rho_0) - 1/(1 + \rho_0)][(3/2)(A_{P1} + B_{P1} + A_{P2}) + (1/2)(H_{P2} + T_{SP} + B_{P0})] \tag{C.225}$$

Appendix D. Analysis of the Schedule Update protocol

The purpose of the Schedule Update mode is to determine the number of messages to be broadcast during the PE Communication mode. The PEs are expected to have agreement on their **desired schedule** before the start of the Schedule Update mode. In this mode, the PEs download to the ROBUS their agreed-upon schedule, in effect, to program the bus. For a system with N BIUs, each PE sends N consecutive messages to its attached BIU with the position in the sequence corresponding to the identification number of the PE to be scheduled and the content of the message specifying the desired number of scheduled messages for that PE. For each of the N PEs to be scheduled, the ROBUS applies the Schedule Update protocol to ensure agreement by the PEs, BIUs, and RMUs on the value received by the bus. After the desired schedule is processed, the ROBUS assesses the received entries and determines whether they form a valid schedule. The result of this assessment is then forwarded to the PEs.

The main goal of the Schedule Update service is to allow properly working PEs to communicate as desired. The most important objective of the Schedule Update protocol is to ensure agreement on each schedule entry even in the presence of faulty PEs. Disagreement on the resulting schedule can result in the disintegration of a ROBUS clique during the execution of the schedule in the PE Communication mode. Figure D.1 illustrates the message flow graph for the Schedule Update protocol. The processes from P0 to P2 form the agreement generation phase, and from P2 to P4 form the agreement propagation phase. Section 5 of this document presents the detailed description of the protocol. The protocol is time-triggered and uses synchronous communication. The protocol combines message processing and diagnostics to ensure agreement even if the number of faulty PEs outnumbers the number of properly working PEs.

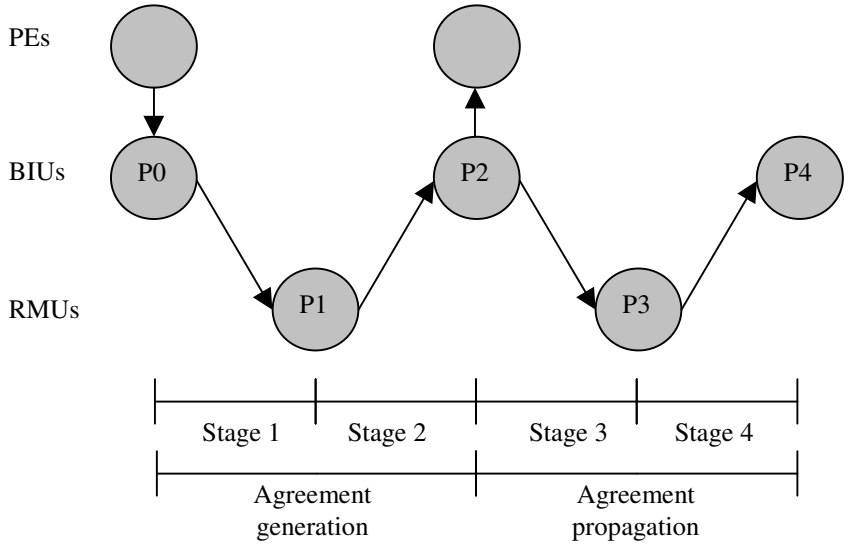


Figure D.1: Message flow graph for the Schedule Update protocol

In process P0, the BIUs serve as relays for the messages from the PEs. Thus, the values received in process P1 and the voting results are dependent on the status of the PEs and the BIUs.

D.1.1. PE classification

Similarly to the BIUs and RMUs, the performance of the PEs depends on whether they are operating according to their specification and their state variables are holding correct values. The PEs are expected to correctly and consistently perform the distributed SPIDER OS functions, including SPIDER-level communication and diagnosis, among others. A group of coordinated PEs that can be relied upon to properly perform these functions is referred to as a **PE clique**. For this version of ROBUS, it is assumed that there is at most one active PE clique at any particular time.

For analysis, the individual PEs are classified according to the following criteria.

- **Goodness**: A PE is **good** if it behaves according to its specification. Otherwise, the PE is **bad** or **faulty**.

- **Trustworthiness**: A PE is **trustworthy** if it is suitable to properly perform the SPIDER OS functions. Otherwise, the PE is **untrustworthy**. A trustworthy PE must be good, and its state must be correct and in agreement with the state of other PE-clique members, if there are any.

A PE clique consists of one or more trustworthy PEs.

D.1.2. PE-BIU pair classification

Because the PEs communicate with the bus through their assigned BIUs, the inputs actually processed by the bus depend on the status of the PEs and the BIUs. A **PE-BIU pair** consists of a PE and its corresponding BIU. Each pair is handled as a unit for the analysis of the input to the Schedule Update protocol.

Each PE-BIU pair is classified according to the following categories.

- **Trustworthy**: A PE-BIU pair in which the PE and the BIU are individually trustworthy.

- **Benign**: An untrustworthy PE-BIU pair in which the PE-BIU pair either broadcasts valid values or is safely removed from eligibility in voting operations.

- **Symmetric**: An untrustworthy PE-BIU pair in which the BIU is symmetric.

- **Asymmetric**: An untrustworthy PE-BIU pair in which the BIU is asymmetric.

D.1.3. Agreement generation phase

The voters for process P1 are the PE-BIU pairs. For proper operation of the protocol, it is required that the trustworthy BIUs in process P0 broadcast PE_ERROR only if their attached PEs are untrustworthy. Thus, a PE_ERROR message is an explicit indication that the PE-BIU pair is untrustworthy. The definition of the protocol and the properties of the diagnostic system ensure that the set of eligible voters in process P1 includes all of the trustworthy PE-BIU pairs.

In what follows, we consider the k-th execution of the Schedule Update protocol, which corresponds to the processing of the desired number of messages for PE k in the schedule computed by the PE clique.

Let $v_{PE,k}$ denote the desired number of messages for PE k. All trustworthy PEs in the PE clique agree on this value.

Let $EV_{P1,i}$ and $EV_{P2,j}$ denote the set of eligible voters for process P1 at RMU i and process P2 at BIU j, respectively. $Twy\_EV_{P1,i}$, $Sym\_EV_{P1,i}$, and $Asym\_EV_{P1,i}$ denote the set of trustworthy, symmetric, and asymmetric eligible voters for process P1 at RMU i, respectively. $Twy\_EV_{P2,j}$, $Sym\_EV_{P2,j}$, and $Asym\_EV_{P2,j}$ denote the set of trustworthy, symmetric, and asymmetric eligible RMU voters for process P2 at BIU j, respectively.

It is assumed that the set of eligible voters for process P2 at the trustworthy nodes contains more trustworthy sources than untrustworthy ones. That is:

$|Twy\_EV_{P2,j}| > |Sym\_EV_{P2,j}| + |Asym\_EV_{P2,j}|$ for process P2 at each trustworthy BIU j.

Following the general protocol theory presented in Appendix A, it is also assumed that in process P1 or process P2 the eligible voters at the trustworthy nodes do not include asymmetric sources. That is:

$|Asym\_EV_{P1,i}| = 0$ for process P1 at each trustworthy RMU i, or

$|Asym\_EV_{P2,j}| = 0$ for process P2 at each trustworthy BIU j.

The following properties hold for the agreement generation phase.

Agreement propagation: If $|Twy\_EV_{P1,i}| > |Sym\_EV_{P1,i}| + |Asym\_EV_{P1,i}|$ for process P1 at each trustworthy RMU i, then the results of the voting operations in process P2 at the trustworthy BIU nodes is equal to $v_{PE,k}$.

Proof: For stage 1, the trustworthy PE-BIU pairs exactly agree on their desired number of messages, and they form a majority of eligible voters at each trustworthy RMU. Therefore, all of the word vote results for process P1 at the trustworthy RMUs will output the value received from the trustworthy PE-BIU pairs, namely $v_{PE,k}$. For stage 2, the trustworthy RMUs exactly agree on the value $v_{PE,k}$, and they form a majority of eligible voters at each trustworthy BIU. Therefore, all of the word vote results for process P2 at the trustworthy BIUs will output the value $v_{PE,k}$.

Agreement generation: The voting results for process P2 at the trustworthy BIU nodes exactly agree.

Proof: Two cases must be considered.

Case 1: $|Asym\_EV_{P1,i}| = 0$: If there are no asymmetric eligible voters in process P1, all of the trustworthy RMUs agree on their set of eligible voters and the corresponding voting inputs. Therefore, the voting results at the trustworthy RMUs exactly agree. The assumption that the trustworthy RMUs form a majority of eligible voters for process P2 at the trustworthy BIUs ensures that the agreement in process P1 propagates to process P2.

Case 2: $|Asym\_EV_{P2,i}| = 0$: If there are no asymmetric eligible voters in process P2, all of the trustworthy BIUs agree on their set of eligible voters and the corresponding voting inputs. Therefore, the voting results exactly agree.

The protocol theory presented in Appendix A requires that, in general, the trustworthy set of eligible

voters for process P1 at the trustworthy nodes contains more trustworthy BIUs than untrustworthy ones. Without the ineligibility of PE_ERROR inputs in process P1, that would indeed be the case. However, since the voters are PE-BIU pairs, and it is known that a PE_ERROR input does not correspond to a valid desired number of messages for PE k, it is not always true that $|Twy\_EV_{P1,i}| > |Sym\_EV_{P1,i}| + |Asym\_EV_{P1,i}|$ at each trustworthy RMU i. Nevertheless, the protocol is able to ensure agreement in process P2, as shown here. Note that a PE-BIU pair can be asymmetric only if the BIU is asymmetric. A PE cannot be asymmetric because it only communicates with its assigned BIU. Therefore, the assumption in the general protocol theory that there are no simultaneous BIU and RMU asymmetric eligible voters in processes P1 and P2 at the trustworthy nodes is the same assumption that ensures agreement generation for the Schedule Update protocol. The protocol ensures agreement in process P2 irrespective of the values submitted by the PEs. It is not even required to have a group of PEs that agree on their values. Only the status of the BIUs and RMUs are relevant in the generation of agreement. However, if it is not true that the trustworthy PE-BIU pairs form a majority of eligible voters in process P1 at each trustworthy RMU, it is possible that the result in process P2 is not the value submitted by any trustworthy PE.

In addition, note that the meaning a PE_ERROR message is different in stage 1 and stage 2. In stage 1, a received PE_ERROR message means that the source PE-BIU pair is untrustworthy. In stage 2, a received PE_ERROR message means that the source RMU determined that there is no agreement among the eligible PE-BIU pairs on the desired number of messages for PE k. A vote result of PE_ERROR in process P2 means that a majority of the eligible RMUs determined that there is no agreement among the PE-BIU pairs, or that the eligible RMUs do not agree on the desired number of messages for PE k. Both conditions are invalid for a schedule update.

Finally, note that disagreement with the result of the vote is not used as an error check in processes P1 or P2. The diagnostic system is required to generate accusations only if it is known with certainty that the accused is untrustworthy. In process P1, it is not possible to determine if a disagreement is caused by the PE or the BIU. In process P2, it is not possible to determine if a disagreement is caused by an untrustworthy RMU or by asymmetric PE-BIU pairs.

D.1.4. Agreement propagation phase

It is assumed that the set of eligible voters for processes P3 and P4 at the trustworthy nodes contains more trustworthy sources than untrustworthy ones. That is:

$|Twy\_EV_{P3,k}| > |Sym\_EV_{P3,k}| + |Asym\_EV_{P3,k}|$ for process P3 at each trustworthy RMU k, and

$|Twy\_EV_{P4,l}| > |Sym\_EV_{P4,l}| + |Asym\_EV_{P4,l}|$ for process P4 at each trustworthy BIU l.

The following property holds for the agreement generation phase.

Agreement propagation: The results of the voting operations for process P3 at the trustworthy RMUs and for process P4 at the trustworthy BIUs are equal to the result for process P2 at the trustworthy BIUs.

Proof: For process P3, exact agreement propagation follows from the fact that the trustworthy BIUs agree on the result for process P2 and they form a majority among the eligible voters in process P3. The conditions are similar for the propagation of agreement from process P3 to process P4.

D.1.5. Schedule assessment

The **received schedule** is the list of results for the N executions of the Schedule Update protocol. The ROBUS examines the received schedule to determine its validity. The **loaded schedule** is the schedule that is accepted by the ROBUS for use in the PE Communication mode. Two schedule validity rules are defined for this version of the ROBUS: (1) none of the schedule entries is equal to PE_ERROR, and (2) the sum of all the schedule entries is less than or equal to the total number of PE messages that can be processed in the PE Communication mode. The received schedule becomes the loaded schedule if it complies with both of these rules. Otherwise, the **default schedule** is loaded.

It is possible that entries in the received schedule are not equal to the corresponding values submitted by trustworthy PEs. The schedule validity rules defined for this version of the ROBUS offer only some protection against the loading of an undesired schedule. The rules can be augmented by appending some sort of check word to the desired schedule list computed by the PEs and comparing the Schedule Update protocol result for the checksum entry against a check word computed by the ROBUS for the received schedule. Increasing the error coverage of the schedule validity rules reduces the likelihood that the loaded schedule does not meet the communication requirements of the PEs.

Appendix E. Analysis of the PE Broadcast and Accusation Exchange protocols

Two different protocols are executed in the PE Communication mode. The PE Broadcast protocol is an agreement protocol with embedded diagnostic processing. This protocol is used in combination with a routing function to ensure that the PE messages are broadcast according to the communication schedule. The Accusation Exchange protocol is intended to enhance the diagnostic capabilities of the ROBUS by allowing fine time granularity for reconfigurations while ensuring that the suspicion-based accusations comply with the required properties of the diagnostic system.

E.1.1. Bus access pattern

The BIUs broadcast PE messages on the BIU-to-RMU links according to the communication schedule. The access pattern is a time-indexed, as-soon-as-possible round-robin in which the first message is sent at a predetermined local time and succeeding messages are sent at regular time intervals. The actual data introduction interval (DII) for the transmissions (denoted by Λ_{stream}) must be larger than the minimum DII of the Communication Module (denoted by Λ_{Comm}) and the minimum DII of the Computation Module (denoted by Λ_{Comp}).

The RMUs receive and route the messages from the BIUs according to the communication schedule. The transmission DII for the RMUs is the same as for the BIUs. The route function ensures that only the scheduled BIU is allowed to access the RMU-to-BIU links. At the receiving end, the BIUs receive and vote on the messages from the RMUs, thus ensuring that failed RMUs will not corrupt the results. Note that, in effect, the output of the RMUs is a stream composed of the individual streams from the scheduled BIU sources.

Most of the time in a diagnostic cycle is available for the broadcast of PE messages. The transmission DII and the processing latency of the PE Broadcast protocol are the main performance determinants of the ROBUS. A reduction in Λ_{stream} and the processing latency increases the total throughput capability of the bus.

E.1.2. PE Broadcast protocol

The PE Broadcast protocol is an interactive consistency protocol based on the generic theory presented in Appendix A and augmented with diagnostic processing capabilities to safely handle transmissions by untrustworthy trusted sources and faulty distrusted sources. The protocol also allows trustworthy nodes to observe and diagnose good recovering nodes by the same means as for trusted nodes. No assumption is made about a relation between the content of the communication schedule and the health and diagnostic status of BIUs.

Figure E.1 illustrates the message flow graph for the protocol. Section 5 of this document presents a detailed description of the protocol. The main purpose of the protocol is to perform a broadcast function in which a message from the source PE is delivered to all the PEs connected to trustworthy BIUs. From the perspective of the ROBUS, the actual content of the message is arbitrary and meaningless. Thus, there is no need for an agreement propagation phase.

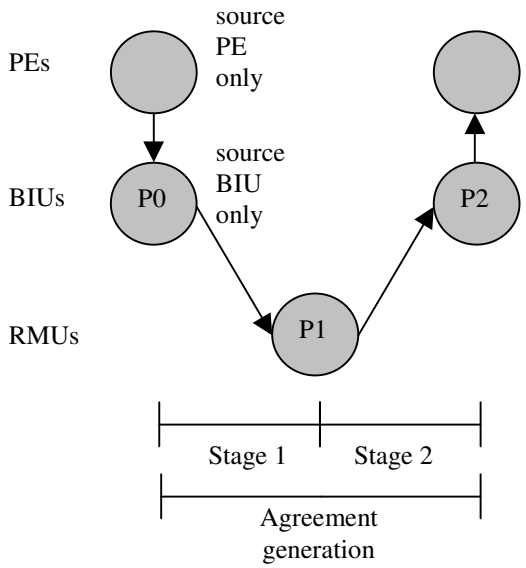


Figure E.1: Message flow graph for the PE Broadcast protocol

Since the Collective Diagnosis protocol ensures agreement among all the trustworthy nodes on the conviction results (see Appendix F), the result sent to the PEs by the trustworthy BIUs for a convicted BIU source is SOURCE_ERROR irrespective of the actual message sent by the source BIU or the voting results at P1 or P2. Likewise, if the trustworthy BIUs agree on accusing the source BIU, the result sent to the PEs will also be SOURCE_ERROR irrespective of the actual voting result in process P2.

We consider the properties of the protocol at the output of the voter in process P2. Irrespective of the health status of the source BIU, it is assumed that the set of eligible voters at each trustworthy BIU contains more trustworthy RMUs than untrustworthy ones. That is:

$|Twy\_EV_{P2,j}| > |Sym\_EV_{P2,j}| + |Asym\_EV_{P2,j}|$ at each trustworthy BIU j.

Validity of the voting result in process P2: If the source BIU is trustworthy, then the result of the vote in process P2 at the trustworthy BIUs is equal to the value sent by the source.

Proof: The source BIU sends the same message to all the RMUs. Since the source is trustworthy, the trustworthy RMUs do not detect input errors or have accusations against it. Therefore, the source BIU is eligible in process P1 and the result at all the trustworthy RMUs is equal to the message sent by the source. Since the trustworthy RMUs are a majority of the eligible voters for process P2 at the trustworthy BIUs, the result of the vote is equal to the value sent by the source BIU.

Agreement on the voting result in process P2: For a given source BIU, if $|Asym\_EV_{P1,i}| = 0$ at each trustworthy RMU i or $|Asym\_EV_{P2,j}| = 0$ at each trustworthy BIU j, then the voting result for process P2 at the trustworthy BIUs exactly agree.

Proof: The source BIU may be asymmetric or otherwise. Two cases are considered.

Case 1: $|Asym\_EV_{P1,i}| = 0$: If the source BIU is asymmetric, then it is not be eligible in process P1 at the trustworthy RMUs and the result is SOURCE_ERROR. In process P2 at the trustworthy BIUs, the messages received from the trustworthy RMUs is a majority of the eligible voters. Therefore, the result of the vote in process P2 is SOURCE_ERROR.

If the source BIU is not asymmetric, then all the trustworthy RMUs agree on whether the source is eligible. If the source is ineligible, the result for process P1 is SOURCE_ERROR. If the source is eligible, the trustworthy RMUs received the same message and the result for process P1 is equal to the received message. Therefore, the trustworthy RMUs agree on the result for process P1. Since the trustworthy RMUs are a majority of eligible voters for process P2 at the trustworthy BIUs, the voting results in process P2 exactly agree.

Case 2: $|Asym\_EV_{P2,j}| = 0$: Since the set of eligible voters for process P2 at each trustworthy BIU does not include asymmetric RMUs, and the diagnostic system is required to satisfy the property of agreement for non-asymmetric defendants, the sets of eligible voters are equal. Since the voting functions will have the same inputs and eligibility set, the voting results in process P2 exactly agree.

The conditions of this agreement property are protocol assumptions for non-convicted sources.

In process P2, the BIUs diagnose the source BIU based on the result of the vote. A vote result of NO_MAJORITY indicates that the source BIU has behaved asymmetrically. A vote result of SOURCE_ERROR indicates that at least one trustworthy RMU considers the source BIU to be untrustworthy. Both of these results are sufficient basis for the BIUs in process P2 to accuse the source BIU. Note that, independently of whether the source BIU is convicted or not, if the agreement property conditions are satisfied, then the trustworthy BIUs in process P2 will agree on their local diagnostic assessment of the source BIU.

If the source BIU is asymmetric and not accused by some trustworthy RMUs, and $|Asym\_EV_{P2,j}| \neq 0$ at some trustworthy BIUs, it is possible not to have agreement on the voting results in process P2 at the trustworthy BIUs. However, if the source BIU is convicted, the PEs will receive the same result of SOURCE_ERROR.

A disagreement with the result of the vote in process P2 indicates that an error occurred somewhere in the path beginning at the source BIU, passing through the disagreeing RMU, and ending at the voting BIU. The detection of disagreement is not sufficient evidence to determine which node is responsible for the error. Because the diagnostic system is required to satisfy the property of correctness, the most that can be done by a receiving BIU in process P2 is to levy a suspicion against the source BIU and the relaying RMU.

E.1.3. Accusation Exchange protocol

The local diagnostic system of each ROBUS node collects accusations and suspicions about nodes of the same kind and the opposite kind. Without the Accusation Exchange protocol, the BIUs would receive information about nodes of their own kind only from the execution of the PE Broadcast protocol, and the RMUs would not receive any information about their own kind. Furthermore, without the Accusation Exchange protocol, the BIUs would be able to observe other BIUs only when (and if) the communication schedule allows it. This constrains the total number of failures that can be tolerated by the bus over a time interval while still maintaining the ability to satisfy the required properties for the generation of

accusations based on the processing of suspicions. The Accusation Exchange protocol allows the processing of suspicions based on the latest available local diagnostic assessments.

Figure E.1 illustrates the message flow graph. Contrary to all of the other protocols, the content of the message broadcast in process P1 is independent of the result computed in that process. The messages in stage 1 are the local accusations by the BIUs against the RMUs, and the messages in stage 2 are the accusations by RMUs against BIUs. Stages 1 and 2 of this protocol are closely related to the processing of SOURCE_ERROR messages in stage 2 of the PE Broadcast protocol.

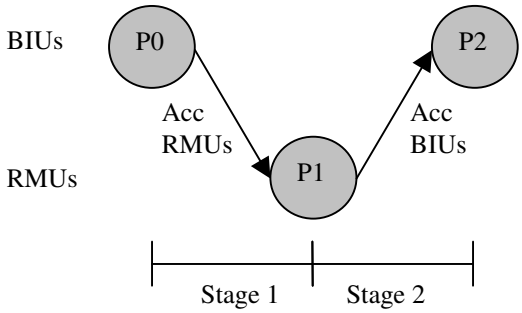


Figure E.1: Message flow graph for the PE Broadcast protocol

For this protocol, it is assumed that the set of eligible voters for processes P1 and P2 at the trustworthy nodes contains more trustworthy sources than untrustworthy ones. That is:

$|Twy\_EV_{P1,i}| > |Sym\_EV_{P1,i}| + |Asym\_EV_{P1,i}|$ for process P1 at each trustworthy RMU i, and

$|Twy\_EV_{P2,j}| > |Sym\_EV_{P2,j}| + |Asym\_EV_{P2,j}|$ for process P2 at each trustworthy BIU j.

We consider stage 2. Similar properties hold for stage 1.

Accusation correctness: For each BIU defendant, the result of the bit vote in process P2 at the trustworthy BIUs is TRUE only if the defendant is indeed untrustworthy.

Proof: The correctness property of the accusation generation mechanisms ensures that the trustworthy RMUs do not accuse trustworthy defendants. Since the trustworthy RMUs form a majority of eligible voters for process P2 at each trustworthy BIU, the result of the bit vote for these defendants is FALSE.

Agreement for non-asymmetric BIU defendants: For each non-asymmetric BIU defendant, the voting results for P2 at the trustworthy BIUs exactly agree.

Proof: If the BIU defendant is non-asymmetric, the trustworthy RMUs agree on the value of their accusation variables for the defendant. Since the trustworthy RMUs are a majority among the eligible voters in process P2, the vote results for the defendant at the trustworthy BIUs agree.

Since the broadcast accusations are the result of diagnosis based on all the observations up to the time of the transmission, agreement on the voting results for process P2 is possible even is the defendant is asymmetric. The following property captures this.

Agreement for a generic BIU defendant: If the trustworthy RMUs agree on the value of their accusation variables for the defendant or $|Asym\_EV_{P2,j}| = 0$ at each trustworthy BIU j, then the trustworthy BIUs agree on the voting result for the defendant in process P2.

Proof: Two cases are considered. If the trustworthy RMUs agree on the value of their accusation variables for the defendant, then the assumption that the trustworthy RMUs form a majority of eligible voters in process P2 ensures that the vote results for the defendant agree.

If the eligible voters for process P2 at the trustworthy BIUs do not include asymmetric RMUs, the BIUs have the same sets of received messages and eligible voters. Therefore, their voting results for the defendant agree.

It is assumed that the conditions of this property are satisfied for non-convicted BIUs defendants. These conditions are essentially the same as the agreement conditions for the PE Broadcast protocol.

Given that the inputs to the bit voter are Boolean (i.e., the value are TRUE or FALSE), if the number of eligible voters is odd, there is always an exact-match majority among the inputs to the bit voter. If the number of eligible voters is even and there is not an exact-match majority among the voter inputs, then half of the inputs are TRUE and the other half are FALSE. Since it is assumed that the trustworthy RMUs form a majority among the eligible voters, then at least one trustworthy RMU accused the defendant. Therefore, a bit vote result of TRUE (i.e., accused) in this case conforms to the required property of accusation correctness. This is similar to the NO_MAJORITY result in stage 2 of the PE Broadcast protocol.

In addition, similarly to the PE Broadcast protocol, a disagreement with the result of a bit vote in process P2 for a particular BIU defendant indicates that an error occurred somewhere in the path from the defendant, through the disagreeing RMU, to the voting BIU. This observation is sufficient evidence to raise suspicions against the defendant and the disagreeing RMU, but it is not enough to accuse either of them. If the result of the bit vote is TRUE, then it is known that the defendant is untrustworthy but it does not necessarily excuse the disagreeing RMU.

Appendix F. Analysis of the diagnostic system

This appendix examines various aspects of the diagnostic system. The properties of the suspicion-based accusation-generation process are presented. The Collective Diagnosis protocol is analyzed using the generic protocol theory presented in Appendix A. In addition, the characteristics of the clique membership are examined using the known properties of the diagnostic system.

The diagnostic system has two accusation generation mechanisms: immediate and suspicion-based. The **immediate accusations** are based on error detection for which there is only one possible culprit, other than the observer itself. **Suspicion-based accusations** are based on the detection of errors with multiple possible culprits and the processing of accumulated suspicions to identify untrustworthy nodes based on accumulated observations. The immediate-accusation mechanisms are known by design to comply with the required properties of correctness and agreement for non-asymmetric defendants. This aspect of the diagnostic system is not explored further. The suspicion-based accusation generation process is examined next.

F.1. Suspicion-based accusations

This section examines the properties of the suspicions-based accusations for a given collective diagnostic interval. The expression **not currently convicted** refers to a node that is classified as not convicted during the current interval. A not currently convicted node was not convicted during the execution of the Collective Diagnosis protocol in the immediately preceding collective diagnostic interval. In contrast, a **currently convicted** defendant is a convicted node during the current interval.

Figure F.1 illustrates the suspicions matrix. As described in Section 4 of this document, every node records it suspicions in a two-dimensional matrix in which the rows correspond to the nodes of the same kind as the observing node and the columns corresponds to the nodes of the opposite kind. Θ and Ω denote the number of nodes of the same kind and the opposite kind, respectively. The suspicions-matrix is processed by bit vote operations for each row and column of the matrix.

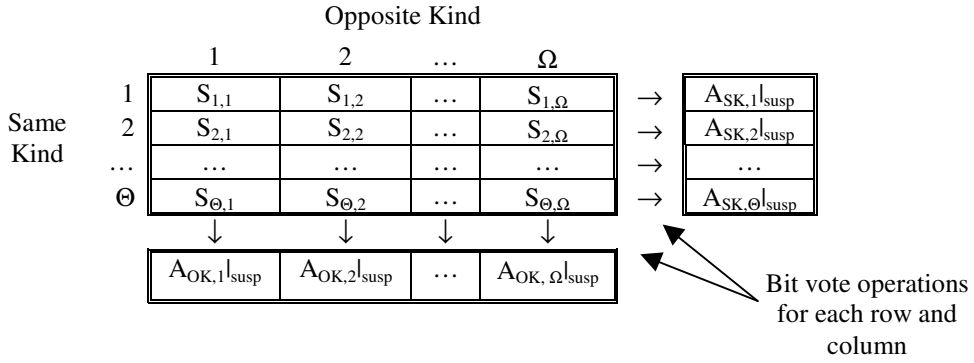


Figure F.1: Suspicions matrix and generated accusations

The suspicions are recorded only during the PE Communication mode, which is the only mode in which the observers gather evidence about the behavior of nodes of their own kind. The BIUs record suspicions based on observations during the execution of the PE Broadcast and Accusation Exchange protocols. The RMUs record suspicions only during the Accusation Exchange protocol. A cell in the suspicions matrix is asserted when there is evidence that one or both of the corresponding nodes are

untrustworthy.

The analysis for the processing at the RMUs is a special case of the analysis for the BIUs. We examine the processing at BIU nodes only.

F.1.1. Processing suspicions against nodes of the opposite kind

The suspicions against nodes of the opposite kind correspond to the columns of the suspicions matrix. These suspicions are processed by a separate bit vote operation for each column. The eligible voters are the trusted nodes of the same kind. Thus, every row corresponding to a distrusted node of the same kind is removed from consideration in the bit vote operations. The Collective Diagnosis protocol ensures agreement among all the trustworthy nodes on the conviction results. In addition, the properties of the PE Communication protocols ensure that, for each not currently convicted node of the same kind, the trustworthy BIUs agree on their accusations performed up to the time at which the suspicions matrix is processed. Therefore, the trustworthy BIUs agree on the set of eligible voters. Furthermore, it is assumed that the set of eligible voters includes more trustworthy than untrustworthy nodes. Let $EV_{SK,i}$ denote the set of eligible voters of the same kind at BIU i. $Twy\_EV_{SK,i}$, $Sym\_EV_{SK,i}$, and $Asym\_EV_{SK,i}$ denote the sets of trustworthy, symmetric, and asymmetric eligible voters, respectively. Thus:

$|Twy\_EV_{SK,i}| > |Sym\_EV_{SK,i}| + |Asym\_EV_{SK,i}|$ for each trustworthy BIU i.

We need to show that the accusation results generated by the bit vote operations satisfy the required properties of correctness and agreement for non-asymmetric defendants.

For a particular defendant of the opposite kind, if the result of the bit vote is TRUE, then, for the suspicions-matrix column corresponding to the defendant, at least one cell corresponding to a trustworthy node of the same kind is TRUE. This means that at least once during the PE Communication mode the value received from the given defendant disagreed with the result of a vote involving a trustworthy node of the same kind. The validity property for the PE Broadcast and Accusations Exchange protocols ensures that the result of the vote is the correct value. Therefore, the given defendant is indeed untrustworthy.

If the defendant is non-asymmetric, the trustworthy BIUs agree on their observations of the defendant. Since the trustworthy BIUs always agree on the vote results in the PE Communication mode involving not currently convicted nodes of their own kind, the trustworthy BIUs agree on the suspicion entries corresponding to the eligible voters. Therefore, the trustworthy BIUs will agree on the bit vote results for the defendant.

F.1.2. Processing suspicions against nodes of the same kind

The suspicions against nodes of the same kind correspond to the rows of the suspicions matrix. The eligible voters for the bit vote operations along the rows are the trusted nodes of the opposite kind. Every column corresponding to a distrusted node of the opposite kind is removed from consideration in the bit vote operations. The properties of the diagnostic system ensure that the trustworthy BIUs agree on their accusations for non-asymmetric nodes of the opposite kind. Furthermore, it is assumed that the set of eligible voters includes more trustworthy than untrustworthy nodes. Let $EV_{OK,i}$ denote the set of eligible voters of the opposite kind at BIU i. $Twy\_EV_{OK,i}$, $Sym\_EV_{OK,i}$, and $Asym\_EV_{OK,i}$ denote the sets of trustworthy, symmetric, and asymmetric eligible voters, respectively. Thus:

$|Twy\_EV_{OK,i}| > |Sym\_EV_{OK,i}| + |Asym\_EV_{OK,i}|$ for each trustworthy BIU i.

For a particular defendant of the same kind, if the result of the bit vote is TRUE, then, for the suspicions-matrix row corresponding to the defendant, at least one cell corresponding to a trustworthy node of the opposite kind is TRUE. This means that the at least once during the PE Communication mode the value received from a trustworthy node of the opposite kind disagreed with the result of a vote involving the defendant. The message transmitted by a trustworthy RMU in the PE Broadcast and Accusation Exchange protocols are a direct result of the messages received from the defendant and the local diagnostic assessment. Therefore, the node responsible for the disagreements is the defendant, which makes it untrustworthy. Thus, the property of correctness is satisfied.

If the defendant is non-asymmetric, the trustworthy RMUs agree on their observations and local diagnosis of the defendant. Furthermore, since the trustworthy RMUs considered for the processing of suspicions are always part of the majority among the eligible voters in the protocols of the PE Communication mode, the values received from them never disagree with the result of the vote. Therefore, for the suspicions-matrix row corresponding to the defendant, the cells corresponding to the trustworthy RMUs are FALSE. Consequently, the bit vote result will be FALSE and no new accusations are raised against the defendant. Since the trustworthy BIUs always agree on their local diagnostic assessment of trustworthy RMUs, a non-asymmetric defendant of the same kind will not be accused in any of them as a result of processing the suspicions matrix.

Consider a not-currently-convicted defendant of the same kind. The properties of the PE Communication protocols ensure that the trustworthy BIUs agree on their vote results for the given kind of defendant. Therefore, for a not-currently-convicted defendant, the trustworthy BIUs agree on the suspicions-matrix entries corresponding to non-asymmetric nodes of the opposite kind. The bit vote results at the trustworthy BIUs agree if the suspicions-matrix entries corresponding to the trustworthy nodes of the opposite kind agree or there are no asymmetric eligible voters at any of the trustworthy BIUs. To show this, we consider the two conditions separately. If the first condition were true, then the bit vote results at the trustworthy BIUs would agree because the trustworthy RMUs form a majority among the eligible voters. If the second condition were true, the bit vote results would agree because there would be agreement on the inputs to the vote and the eligible voters.

Note that if the required conditions for the agreement properties of the PE Communication protocols hold for a currently convicted defendant of the same kind, then the processing of suspicions for that defendant has the same properties as for a not currently convicted one.

F.2. Collective Diagnosis protocol

The on-line diagnosis protocol was developed by Geser and Miner. The formal verification of the protocol is presented in [Geser 04].

This section examines the Collective Diagnosis protocol for BIU defendants. The analysis of the protocol for the diagnosis of RMU defendants is similar. Figure F.2 illustrates the message flow graph for the Collective Diagnosis protocol for BIU defendants. The protocol processes are from P0 to P4. The actual protocol processes multiple defendants in parallel. To simplify the presentation, we examine the characteristics of the protocol for a single defendant. The “def” bubble represents the defendant, and the dashed arrow represents the diagnostic evidence collected by the RMU nodes and used to generate the accusations against the defendant. The RMUs are direct observers of the defendant, and the BIUs are

indirect observers. Every ROBUS protocol provides an opportunity for the RMUs to collect evidence against the defendant. The BIUs are able to observe the defendant only in the PE Communication mode. If the defendant is a scheduled source, the PE Broadcast protocol enables the BIUs to observe the messages broadcast by the defendant and relayed by the RMUs after some processing. The Accusation Exchange protocol enables the BIUs to observe the defendant only in terms of the accusations submitted by the RMUs.

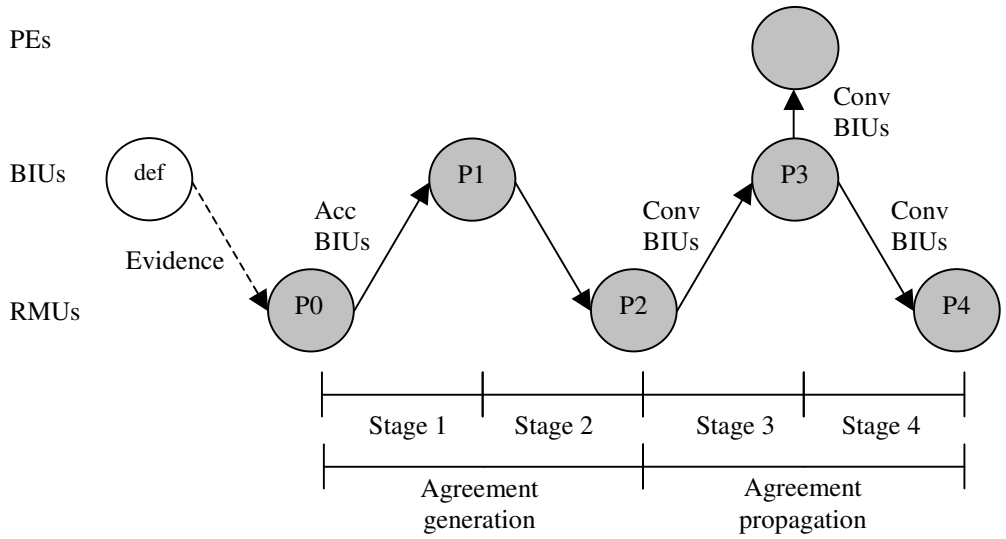


Figure F.2: Message flow graph for the Collective Diagnosis protocol for BIU defendants

It is assumed that each of the sets of eligible voters for processes P1 through P4 at the trustworthy nodes contains more trustworthy sources than untrustworthy ones. That is:

$|Twy\_EV_{P1,i}| > |Sym\_EV_{P1,i}| + |Asym\_EV_{P1,i}|$ for process P1 at each trustworthy BIU i,

$|Twy\_EV_{P2,j}| > |Sym\_EV_{P2,j}| + |Asym\_EV_{P2,j}|$ for process P2 at each trustworthy RMU j,

$|Twy\_EV_{P3,k}| > |Sym\_EV_{P3,k}| + |Asym\_EV_{P3,k}|$ for process P3 at each trustworthy BIU k, and

$|Twy\_EV_{P4,l}| > |Sym\_EV_{P4,l}| + |Asym\_EV_{P4,l}|$ for process P4 at each trustworthy RMU l.

In addition, it is also assumed that in process P1 or process P2 the eligible voters at the trustworthy nodes do not include asymmetric sources. That is:

$|Asym\_EV_{P1,i}| = 0$ at each trustworthy BIU i, or

$|Asym\_EV_{P2,j}| = 0$ at each trustworthy RMU j.

F.2.1. Agreement generation phase

We begin the analysis of the agreement generation phase by considering conviction agreement for a not-currently-convicted defendant. The properties of the PE Communication protocols and of the

accusation generation mechanisms, including the processing of the suspicions matrix, ensure that the trustworthy BIUs have agreement on the accusations generated against the defendant.

Agreement for not-currently-convicted defendants: If the defendant is not currently convicted, the trustworthy RMUs agree on the voting results for process P2.

Proof: Two cases are considered.

Case 1: $|Asym\_EV_{P1,i}| = 0$: Since all of the eligible voters for process P1 at the trustworthy BIUs are non-asymmetric, they all agree on the voting inputs and the eligible voters. Therefore, they agree on the result of the bit vote. Since there is agreement on the bit vote result and on the local accusation against the defendant, the merge operation, a simple Boolean OR function, preserves that agreement. Stage 2 is an agreement propagation stage, and thus the trustworthy RMUs agree on the result of the bit vote for process P2.

Case 2: $|Asym\_EV_{P2,j}| = 0$: For this case, the trustworthy RMUs agree on the eligible voters and the inputs received from them for process P2. Therefore, they also agree on the bit vote result.

Next, consider a currently convicted defendant. In general, there is no guarantee of agreement among the trustworthy BIUs on the local diagnosis of a currently convicted BIU.

Agreement for currently convicted defendants: If the defendant is currently convicted, and the condition that $|Asym\_EV_{P1,i}| = 0$ at each trustworthy BIU i implies that the trustworthy BIUs agree on the accusations against the defendant, then the trustworthy RMUs agree on the voting results for process P2.

Proof: Two cases are considered.

Case 1: $|Asym\_EV_{P1,i}| = 0$: For this case, the trustworthy BIUs agree on the eligible voters and the inputs received from them for process P1. Thus, they agree on the result of the bit vote. By the premises, the trustworthy BIUs agree on the accusations against the defendant. The merge operation preserves that agreement. Stage 2 is an agreement propagation stage, and thus the trustworthy RMUs agree on the result of the bit vote for process P2.

Case 2: $|Asym\_EV_{P2,j}| = 0$: For this case, the trustworthy RMUs agree on the eligible voters and the inputs received from them for process P2. Therefore, they also agree on the bit vote result.

Conviction correctness holds irrespective of whether the defendant is currently convicted or not.

Conviction correctness: The result of the bit vote in process P2 is TRUE only if the defendant is untrustworthy.

Proof: Since the trustworthy BIUs are a majority among the eligible voters in process P2, a bit vote result of TRUE implies that a value of TRUE was received from at least one trustworthy BIU. The result of the merge operation in process P1 at a trustworthy BIU is TRUE if the local accusation is TRUE or the result of the bit vote is TRUE. The properties of the accusation generation mechanisms ensure that the local accusations are correct. If the result of the bit vote is TRUE, then at least one value of TRUE was received from process P0 at a trustworthy RMU. The properties of the accusation generation mechanisms ensure that the accusations at the trustworthy RMUs are correct. Therefore, the defendant is convicted only if it has been accused by a trustworthy observer.

For a BIU defendant, the Collective Diagnosis protocol guarantees convictions for the following cases.

- The defendant is benign or symmetric untrustworthy and accused by BIUs or RMUs.

- The defendant is asymmetric untrustworthy and accused by: (1) a subset of trustworthy RMUs that is at least half of the eligible voters for process P1 at each trustworthy BIU, or (2) a subset of trustworthy BIUs that is at least half of the eligible voters for process P2 at each trustworthy RMU.

F.2.2. Agreement propagation phase

The agreement generated in the first phase is propagated in the second one irrespective of the status of the defendant.

Agreement propagation: The results of the voting operations for process P3 at the trustworthy BIUs and for process P4 at the trustworthy RMUs are equal to the results for process P2 at the trustworthy RMUs.

Proof: For process P3, exact agreement propagation follows from the fact that the trustworthy RMUs agree on the result for process P2 and they form a majority among the eligible voters in process P3. The conditions are similar for the propagation of agreement from process P3 to process P4.

F.3. Clique membership

For a given node, the set of trusted nodes of the same kind and the set of trusted nodes of the opposite kind constitute the node's view of the clique membership. The required properties of correctness and agreement for non-asymmetric defendants establish basic constraints on the trusted sets. Correctness of diagnosis ensures that the trustworthy nodes trust one another. Thus, at each trustworthy node, the trusted set includes all of the trustworthy nodes of the same kind and the opposite kind. This is a basic requirement for maintaining the unity of the clique. The property of agreement for non-asymmetric defendants ensures that the trustworthy nodes of a particular kind agree on their trust assessment for non-asymmetric defendants. This property is needed to ensure that the protocols are able to generate agreement.

When trustworthy nodes of a particular kind have agreement on the clique membership, their agreement has the following characteristics.

- They agree on trusting all trustworthy nodes, irrespective of whether they are of the same or opposite kind.

- For each non-asymmetric node, they agree on either trusting it or not, irrespective of whether the node is of the same or opposite kind.

- For each not-currently-convicted node of the same kind, they agree on either trusting it or not.

- There is no certainty of agreement for asymmetric nodes that are of the opposite kind or currently convicted.

When trustworthy nodes of opposing kinds have agreement on clique membership, they agree as follows.

- They agree on trusting all trustworthy nodes, irrespective of whether they are of the same or opposite kind.

- There is no certainty of agreement for untrustworthy nodes of any kind or status.

Appendix G. Analysis of startup and restart

This appendix examines the startup and restart capabilities. The mode transition graph for the ROBUS nodes is presented in Section 2.

Recovery is the process of reaching the Clique Preservation mode from a disabled or failed state. Startup is a recovery triggered by a power-on enable. Restart is triggered by the detection of a local failure or a bus failure. The recovery process involves the following sequence of steps: reset, self-test, search for an active clique, and attempt to join or form a clique. Four basic recovery cases are defined based on the trigger and whether there is a clique present during recovery. Table G.1 illustrates the relation among the recovery cases. The recovery trigger can be a power-on enable or the detection of a local failure or a bus failure. A **clique-join** case is a recovery with a clique already active on the bus. There is no active clique for a **clique-initialization** case.

		Trigger	
Active Clique Present		**Power-On Enable**	**Failure Detection**
	Yes	Join	Rejoin
	No	Initialization	Re-initialization

Table G.1: Recovery cases

G.1. Recovery limitations

The trustworthy nodes enter the Clique Preservation mode synchronized and with agreement on the diagnostic state. Highly deterministic time-triggered behavior in this mode enables it to have substantially robust fault-tolerance. The Clique Join mode shares similar advantages. Other major modes are less robust.

The most important characteristic for the Self-Test mode is the coverage of the test. Ideally, the coverage is sufficiently high to detect most faults, especially permanent faults, and the nodes are implemented in such a way that a node will not exit this mode unless it is fault-free. Such behavior essentially corresponds to a fail-stop on recovery, which increases the chances that other good recovering nodes will successfully reach the Clique Preservation mode.

The Clique Detection mode consists of attempting to acquire the state of a clique while simultaneously monitoring for the absence of one. The effectiveness of this mode is limited by the ability of recovering nodes to correctly diagnose observed nodes without referencing its local state or comparing messages received from different nodes. The error detection and diagnosis mechanisms are less effective without these references. In addition, accusations asserted during the Clique Detection mode could last two or three times the duration of accusations made in the Clique Preservation mode. This can result in scenarios in which trustworthy nodes newly arrived to a clique are accused. Thus, there is a chance of false positive and false negative diagnoses in the Clique Detection mode. If a clique is present, successful diagnosis and state acquisition requires that a sufficient number of untrustworthy nodes are removed from the trusted set such that a majority of trusted nodes are trustworthy. Successful detection that no clique is present during recovery requires that the recovering node correctly diagnose that there are no trustworthy nodes of one kind or another. The Clique Detection mode as presented in Section 7 has the special

vulnerability that, if the set of eligible voters does not have more trustworthy nodes than untrustworthy ones, the Synchronization Capture protocol may not correctly synchronize the local time and may not even finish at all. Such a violation of the protocol assumptions can occur not only due to the inability of Local Diagnosis Acquisition and Frame Synchronization to identify a sufficient number of untrustworthy nodes, but also if there is a decrease in the number of trustworthy clique members after Synchronization Capture is started.

It is a requirement that a group of recovering nodes must have a relative local-time skew that remains within a known bound during Initial Diagnosis and Initial Synchronization. Compliance with the bounded-skew requirement enables the nodes to execute Initial Diagnosis with synchronous communication even if the skew bound is large. The specification of the time interval for reception, the number of expected messages, and the message content for Initial Diagnosis enhances the effectiveness of its error detection and diagnosis. The success of Initial Synchronization is dependent on the validity of the actual local-time skew and the eligible voter sets. Similarly to the Synchronization Capture protocol in the Clique Detection mode, the Initial Synchronization protocol presented in Section 9 has the vulnerability that a violation of the protocol assumptions can result in an incorrectly synchronized local time or a state of indefinite suspended activity.

The limitations of the diagnostic system in the Clique Detection and Clique Initialization modes, and the requirement of having a known bound on the relative local-time skew in the Clique Initialization mode constrain the effectiveness of the ROBUS recovery scheme. Providing a comprehensive design that ensures a successful recovery for all possible scenarios is beyond the scope of this ROBUS version. The ROBUS recovery capability is intended to handle scenarios of independent and non-overlapping recovery cases. For example, proper handling of scenarios in which a set of good nodes is initializing a new clique while another good node is simultaneously trying to join a clique are not considered essential for the design. Although there are circumstances for which such scenarios eventually result in a clique that includes all of the recovering nodes, in general, the recovery cases are only considered to occur separately. The robust diagnostic capabilities of the ROBUS nodes should allow them to detect a failed recovery and initiate a retry, thus increasing the chances of successful recovery. Analysis of the success rate for all possible recovery scenarios is beyond the scope of this appendix.

The response of the ROBUS nodes for the previously mentioned failure modes of the Synchronization Capture and Initial Synchronization protocols can be improved by adding a pair of error checks. The first one is a timeout check to ensure that the nodes eventually exit out of the synchronization protocols even if the assumptions are not satisfied. A timeout check resource can also be used for a check on the resynchronization period in the Clique Preservation and Clique Join modes. The second additional check for Synchronization Capture and Initial Synchronization is a comparison between the number of eligible voters at the beginning of the protocol and the number of eligible voters at the end. This requires the execution of an error detection and diagnosis activity in parallel with the synchronization protocols. For the Synchronization Capture protocol, this diagnostic activity can be realized by continuing the Local Diagnosis Acquisition checks during Synchronization Acquisition. For the Initial Synchronization protocol, the diagnostic activity may consist of custom checks based on the expected message pattern for each opposite kind source during this protocol. For Synchronization Acquisition and Initial Synchronization, it is assumed that the number of trustworthy eligible voters is greater than the number of untrustworthy ones, and that the trustworthy voters will remain so during the execution of the protocols. If fewer than a majority of the initially eligible voters satisfy the eligibility conditions at the end of the protocol, then the protocol assumption have been violated and a failure detection can be asserted.

The triggering of recovery is modeled as discrete distributed events that involve a particular set of

nodes where recovery is triggered during a given finite time interval. Each instance of these discrete distributed events is referred to as a **recovery-trigger event** and is characterized by a precision and a set of recovering nodes. The relative skew of a recovery trigger between two recovering nodes is the real time elapsed between the triggering of the recovery at the two nodes. The precision of a recovery-trigger event is the largest relative skew of the recovery trigger for the set of recovering nodes. The **recovery process** for a particular recovery-trigger event is the activity on the bus from the start of the event until the completion of the recovery.

The precision of a recovery-trigger event is an important determinant of the relative local-time skew during Initial Diagnosis and Initial Synchronization in the Clique Initialization mode. Other relevant factors are the duration of the Self-Test and Clique Detection major modes, and the duration of the Initial Diagnosis and Initial Synchronization minor modes. Once a recovery is triggered, a certain amount of time is required for the bus to return to normal steady-state operation. The time between recovery-trigger events should be large enough so the recovery process of one event is complete before the next event arrives. For the design and analysis of the ROBUS recovery scheme, it is assumed that there is a known bound for the precision of the recovery-trigger events, and that the events occur sufficiently apart so that the recovery processes are effectively independent and non-overlapping.

G.2. Clique initialization

We examine the timing aspects of the clique-initialization recovery cases, which include initialization with a power-on enable recovery trigger and re-initialization triggered by the detection of a failure. For these cases, a clique is not present on the bus during recovery, and thus the major mode path through the Clique Initialization mode is followed to reach the Clique Preservation mode.

G.2.1. Power-one enable

It is assumed that the nodes enter the Self-Test mode immediately after power-on enable. δ_{POE} denotes the actual duration of the time interval within which the nodes are enabled, measured in units of seconds. $\delta_{POE}|_{max}$ denotes the upper bound for δ_{POE}. π_{POE} denotes the upper bound on the relative time skew at power-on enable, measured in nominal clock ticks. τ_0 denotes the nominal duration of a clock tick measured in seconds. δ_{POE} and π_{POE} are related as follows:

$$\pi_{POE} = \delta_{POE}|_{max}/\tau_0 \qquad \text{(G.1)}$$

G.2.2. Local failure or bus failure

δ_{FCP} denotes the actual duration of a fault-causing phenomenon measured in units of seconds. We assume that the duration of the fault-causing phenomenon as experienced by individual nodes can be effectively 0. (Note that $\delta_{FCP} = 0$ means that the phenomenon has a negligibly small duration, not that the phenomenon has no effect.) So:

$$\delta_{FCP}|_{min} = 0 \qquad \text{(G.2)}$$

$\delta_{FCP}|_{max}$ depends on the characteristics of the fault-causing phenomenon for which the design is targeted.

Δ_{FD} denotes the actual duration of the failure-detection delay measured in local clock ticks. We

assume that it is possible for a node to detect a failure condition immediately. $\Delta_{FD}|_{max}$ is implementation-dependent. δ_{FD} denotes the actual duration of the failure-detection delay measured in nominal clock ticks.

$$\delta_{FD}|_{min} = 0 \tag{G.3}$$

$$\delta_{FD}|_{max} = (1 + \rho_0)\Delta_{FD}|_{max} \tag{G.4}$$

Let $t_{FCP,0}$ denote the time at which the fault-causing phenomenon begins. Let $t_{restart,l}$ and $t_{restart,h}$ denote the earliest and latest times, respectively, at which nodes affected by the fault-causing phenomenon enter the Self-Test mode.

$$t_{restart,l} = t_{FCP,0} + \delta_{FCP}|_{min} + \delta_{FD}|_{min}$$

$$= t_{FCP,0} \tag{G.5}$$

And:

$$t_{restart,h} = t_{FCP,0} + \delta_{FCP}|_{max}/\tau_0 + (1 + \rho_0)\Delta_{FD}|_{max} \tag{G.6}$$

Let $\pi_{restart}$ denote the upper bound on the relative time skew when entering the Self-Test mode for restart, measured in nominal clock ticks.

$$\pi_{restart} = t_{restart,h} - t_{restart,l}$$

$$= \delta_{FCP}|_{max}/\tau_0 + (1 + \rho_0)\Delta_{FD}|_{max} \tag{G.7}$$

G.2.3. Self-Test mode

We want to determine the duration of the Self-Test mode and the bound on the relative skew upon exiting this mode.

G.2.3.1. Duration of the Self-Test mode

The **Local Upset Abatement Delay** (LUAD) for a transient-fault scenario is defined as the delay from the time the fault-causing phenomenon reaches a node until the node has regained control of its local operation. Local regaining of control is assumed to occur after the node has detected the failure condition, at which time the node disables its broadcast outputs and transitions to the Self-Test mode. In the Self-Test mode, the node first performs a full local reset and then begins the execution of the self-test procedure. This local reset activity should cover the Communication and Computation modules. The duration of the reset is implementation-dependent. Let δ_{LUAD} denote the actual duration of the Local Upset Abatement Delay, measured in units of nominal clock ticks.

$$\delta_{LUAD}|_{max} = t_{restart,h} - t_{FCP,0}$$

$$= \delta_{FCP}|_{max}/\tau_0 + (1 + \rho_0)\Delta_{FD}|_{max} \tag{G.8}$$

The **Observed Upset Abatement Delay** (OUAD) for a transient-fault scenario is defined as the delay from the time the fault-causing phenomenon begins until the affected nodes can be consistently

recognized by all direct observers as being untrustworthy. Note that this delay is defined with respect to the effects perceived by the observing nodes. Thus, the reception delay must be taken into consideration. Let δ_{OUAD} denote the actual duration of the Observed Upset Abatement Delay, measured in units of nominal clock ticks.

$$\delta_{OUAD}|_{max} = \delta_{LUAD}|_{max} + r_{PP,h}$$

$$= \delta_{FCP}|_{max}/\tau_0 + (1 + \rho_0)\Delta_{FD}|_{max} + r_{PP,h} \quad \text{(G.9)}$$

Δ_{STM} denotes the duration of the Self-Test mode for a ROBUS node, measured in units of local clock ticks. Δ_{STM} is assumed constant. The duration of the Self-Test mode must satisfy the timing requirements for the expected transient-fault scenarios. To increase the probability that a restarting node does not trust an affected node, we require that the restarting nodes exit the Self-Test mode only after the latest time at which affected nodes can be recognized as untrustworthy. So:

$$t_{restart,l} + \Delta_{STM}/(1 + \rho_0) \geq t_{restart,h} + r_{PP,h}$$

$$\Delta_{STM}/(1 + \rho_0) \geq \pi_{restart} + r_{PP,h}$$

$$\Delta_{STM}/(1 + \rho_0) \geq \delta_{OUAD}|_{max} \quad \text{(G.10)}$$

In terms of local clock ticks, the above inequality corresponds to the following constraint:

$$\Delta_{STM} \geq \lceil (1 + \rho_0)\delta_{OUAD}|_{max} \rceil \quad \text{(G.11)}$$

G.2.3.2. Bound on the relative local-time skew at the end of the Self-Test mode

π_{STM} denotes the upper bound on the relative time skew at the end of the Self-Test mode, measured in nominal clock ticks. So:

$$\pi_{STM} = \max(\pi_{POE}, \pi_{restart}) + [(1 + \rho_0) - 1/(1 + \rho_0)]\Delta_{STM} \quad \text{(G.12)}$$

G.2.4. Clique Detection mode

The Clique Detection mode is composed of three main operations: Local Diagnosis Acquisition, Synchronization Acquisition, and Collective Diagnosis Acquisition.

G.2.4.1. Local Diagnosis Acquisition

Local Diagnosis Acquisition is composed of two consecutive observation intervals, each with a duration at least as large as a resynchronization interval. $\Delta_{LDA,begin}$ denotes the delay from the time a node exits the Self-Test mode until the beginning of the first observation interval, measured in local-clock ticks. The value of $\Delta_{LDA,begin}$ is determined by the implementation and assumed constant.

G.2.4.1.1. Bound on the duration of an observation phase

It is assumed that, at the earliest, a node in Local Diagnosis Acquisition can detect the absence of a valid clique as soon as it enters the observation phase. $\Delta_{LDA,OW}$ denotes the duration of the observation intervals (or "windows"), measured in local-clock ticks. The value of $\Delta_{LDA,OW}$ is determined by the implementation and assumed constant. $\Delta_{LDA,OW}$ should be large enough to cover the duration of a resynchronization cycle measured in local-clock ticks. So:

$$\Delta_{LDA,OW} \geq P \quad \text{(G.13)}$$

P is given in Appendix C.

G.2.4.1.2. Bound on the duration of Local Diagnosis Acquisition

Let δ_{LDA} denote the actual duration of Local Diagnosis Acquisition measured in nominal clock ticks.

$$\delta_{LDA}|_{min} = \Delta_{LDA,begin}/(1+\rho_0) \quad \text{(G.14)}$$

$$\delta_{LDA}|_{max} = (1+\rho_0)(\Delta_{LDA,begin} + 2\Delta_{LDA,OW}) \quad \text{(G.15)}$$

G.2.4.2. Synchronization Acquisition

Synchronization Acquisition is composed of the Frame Synchronization and Synchronization Capture protocols. Synchronization Acquisition ends with the synchronization reset, at which point the local time is set to 0.

G.2.4.2.1. Frame Synchronization

It is assumed that a node can detect the absence of a valid clique at any time during Synchronization Acquisition. $\Delta_{FS,begin}$ denotes the delay from the end of the second observation window during Local Diagnosis Acquisition to the beginning of the Frame Synchronization protocol during Synchronization Acquisition, measured in local clock ticks. $\Delta_{FS,begin}$ is implementation-dependent and assumed constant.

Δ_{FS} denotes the actual duration of the execution of the Frame Synchronization protocol measured in local clock ticks. Δ_{FS} is given in Appendix C.

δ_{FS} denotes the actual duration of the Frame Synchronization protocol measured in nominal clock ticks.

$$\delta_{FS}|_{max} = (1 + \rho_0)\Delta_{FS}|_{max} \quad \text{(G.16)}$$

G.2.4.3. Synchronization Capture

We assume that the Synchronization Capture protocol begins immediately after the Frame Synchronization protocol is complete.

G.2.4.3.1. Bound on the duration of the Synchronization Capture protocol

δ_{SC} denotes the actual duration of the execution of the Synchronization Capture protocol measured in nominal clock ticks. The execution of the protocol may begin shortly after the ECHO messages are transmitted by the clique during the execution of the Synchronization Preservation protocol. In that case, the end of the Synchronization Capture protocol would occur after the reset is applied during the next execution of the Synchronization Preservation protocol. To specify a bound for the duration of the Synchronization Capture protocol, we consider an interval containing two consecutive executions of the Synchronization Preservation protocol. $\delta_{SP}|_{max}$ denotes the upper bound on the real-time duration of the execution of the Synchronization Preservation protocol. $\delta_{SP}|_{max}$ is given in Appendix C. T_{SP} denotes the scheduled local time at which the execution of the Synchronization Preservation protocol begins.

$$\delta_{SC}|_{max} = p_{max} + \delta_{SP}|_{max}$$

$$= (1 + \rho_0)T_{SP} + 2\delta_{SP}|_{max} \quad \text{(G.17)}$$

G.2.4.4. Bound on the duration of Synchronization Acquisition

Let δ_{SA} denote the actual duration of the execution of the Synchronization Acquisition measured in nominal clock ticks.

$$\delta_{SA}|_{max} = (1+\rho_0)\Delta_{FS,begin} + \delta_{FS}|_{max} + \delta_{SC}|_{max} \quad \text{(G.18)}$$

Δ_{SA} denotes the actual duration of Synchronization Acquisition measured in local clock ticks. We want to ensure that a count of $\Delta_{SA}|_{max}$ local ticks takes no fewer than $\delta_{SA}|_{max}$ nominal ticks.

$$\Delta_{SA}|_{max}/(1+\rho_0) \geq \delta_{SA}|_{max} \quad \text{(G.19)}$$

We choose the minimum value that satisfies the constraint. That is:

$$\Delta_{SA}|_{max} = \lceil (1+\rho_0)\delta_{SA}|_{max} \rceil \quad \text{(G.20)}$$

G.2.4.5. Bound on the duration of the Clique Detection mode

Synchronization Acquisition ends with the synchronization reset, at which point the local time is set to 0. From that point on, the local time should be synchronized to the clique in Preservation mode. The delays to begin and complete the Collective Diagnosis Acquisition protocol in the Clique Detection mode are the same as for the Collective Diagnosis protocol in Clique Preservation mode. $\Delta_{CD,begin}$ denotes the time from the synchronization reset to the beginning of the Collective Diagnosis protocol, measured in local clock ticks. Δ_{CD} denotes the time to complete the execution of the Collective Diagnosis protocol in local clock ticks. The transition to the Clique Join mode occurs at the beginning of execution of the Schedule Update protocol. Before that point, a detected failure attributable to the absence of a clique results in a transition to the Clique Initialization mode. $\Delta_{SU,begin}$ denotes the time from the end of the Collective Diagnosis protocol to the beginning of the Schedule Update protocol, measured in local clock ticks. $\Delta_{CD,begin}$, Δ_{CD}, and $\Delta_{SU,begin}$ are implementation-dependent and determined by the time-indexed operation schedule specifying the timing for bus activities. Section 3 presents the concept of distributed synchronous composition.

After detecting the absence of a valid clique, a node clears its state and transitions to the Clique Initialization mode. $\Delta_{CDM\text{-}CIM}$ denotes the delay to transition to the Clique Initialization mode after detecting the absence of a valid clique, measured in units of local clock ticks. $\Delta_{CDM\text{-}CIM}$ is implementation-dependent and assumed constant. δ_{CDM} denotes the actual duration of the Clique Detection mode for a ROBUS node, measured in units of nominal clock ticks.

$$\delta_{CDM}|_{min} = \delta_{LDA}|_{min} + \Delta_{CDM\text{-}CIM}/(1 + \rho_0) \quad \text{(G.21)}$$

$$\delta_{CDM}|_{max} = \delta_{LDA}|_{max} + \delta_{SA}|_{max} + [(1+\rho_0)\ (\Delta_{CD,begin} + \Delta_{CD} + \Delta_{SU,begin} + \Delta_{CDM\text{-}CIM})] \quad \text{(G.22)}$$

G.2.4.6. Bound on the relative local-time skew at the beginning of the Clique Initialization mode

$\pi_{CIM,BEGIN}$ denotes the upper bound on the relative time skew at the beginning of the Clique Initialization mode, measured in nominal clock ticks.

$$\pi_{CIM,BEGIN} = \pi_{STM} + (\delta_{CDM}|_{max} - \delta_{CDM}|_{min}) \quad \text{(G.23)}$$

G.2.5. Initial Diagnosis

To simplify the presentation, we would like to compute a single upper bound for the relative local-time skew during the execution of the Initial Diagnosis and Initial Synchronization protocols. Let π_{ID+IS} denote that bound, measured in nominal clock ticks.

For Initial Diagnosis, the BIUs and RMUs are assumed to have the same timing characteristics. The analysis presented here does not refer to the kind of the node sending or receiving messages for any of the protocol processes.

$\Delta_{ID,begin}$ denotes the delay from the time a node enters the Clique Initialization mode until the time it begins the execution of the Initial Diagnosis protocol, measured in units of local clock ticks. The value of $\Delta_{ID,begin}$ is determined by the implementation and assumed constant.

G.2.5.1. Communication between processes P0 and P1

The following variables are defined: T_{ID} denotes the local time triggering the execution of the Initial Diagnosis protocol; $T_{ID,P0\text{-}P1,REF}$ denotes the reference time for the communication between processes P0 and P1; $T_{ID,P0,SND}$ denotes the time at which process P0 sends the message; $T_{ID,P1,RCV,E}$ denotes the expected time of reception in process P1; $S_{ID,P0}$ denotes the Send Process delay for process P0; $\Delta_{ID,P1,RCVWND}$ denotes the delay from the communication reference time to the opening of the receive window in process P1; R_{PP} denotes the nominal point-to-point reception delay; $W_{ID,Deskew}$ denotes the size of the deskewing window in process P1; $W_{ID,Deskew,pre}$ denotes the pre-expectation window in process P1 (i.e., the size of the section of the deskewing window before the expected time of reception); $W_{ID,Deskew,post}$ denotes the post-expectation window in process P1 (i.e., the size of the section of the deskewing window after the expected time of reception); $\Delta_{ID,PP,RCV}|_{abs\text{-}max}$ denotes the absolute value of the maximum error in the actual time of reception in process P1 for a good source-receiver pair; $C_{ID,P1}$ denotes the computation delay in process P1 (The computation delay is measured from the end of the deskewing window. $C_{CD,P1}$ is assumed constant.); $\Delta_{ID,P1,C\text{-}END}$ denotes the delay in process P1 from the end of the computation to the end of the execution of the Initial Diagnosis protocol ($\Delta_{ID,P1,C\text{-}END}$ is assumed constant.); and Δ_{ID} denotes the

duration of the execution of the Initial Diagnosis protocol.

T_{ID} is the reference time for the communication between processes P0 and P1. Given that the local time is reset at the start of the Clique Initialization mode, then:

$$T_{ID,P0\text{-}P1,REF} = T_{ID} = \Delta_{ID,begin} \tag{G.24}$$

To determine $W_{ID,Deskew}$, we need the maximum error in the expected time of reception for the Initial Diagnosis protocol messages, $\Delta_{ID,PP,RCV}|_{abs\text{-}max}$. Based on the analysis presented in Appendix B for point-to-point communication:

$$\Delta_{ID,PP,RCV}|_{abs\text{-}max} = \lfloor (1 + \rho_0)(\pi_{ID+IS} + \max(\mu_{PP,l} , \mu_{PP,h})) \rfloor \tag{G.25}$$

$\mu_{PP,l}$ and $\mu_{PP,h}$ are given in Appendix B. So, for the deskewing window:

$$W_{ID,Deskew} = 2\Delta_{ID,PP,RCV}|_{abs\text{-}max} + 1 \tag{G.26}$$

$$W_{ID,Deskew,pre} = \Delta_{ID,PP,RCV}|_{abs\text{-}max} \tag{G.27}$$

$$W_{ID,Deskew,post} = \Delta_{ID,PP,RCV}|_{abs\text{-}max} + 1 \tag{G.28}$$

We expect the upper bound on the relative local-time skew during the execution of the protocol to be much larger than any minimum timing constraints associated with the process of communication. Based on this, we assume that the following condition holds for the communication between processes P0 and P1.

$$S_{ID,P0}|_{min} + R_{PP} < \Delta_{ID,P1,RCVWND}|_{min} + W_{ID,Deskew,pre} \tag{G.29}$$

For this case:

$$S_{ID,P0} = \Delta_{ID,P1,RCVWND}|_{min} + W_{ID,Deskew,pre} - R_{PP} \tag{G.30}$$

And:

$$\Delta_{ID,P1,RCVWND} = \Delta_{ID,P1,RCVWND}|_{min} \tag{G.31}$$

So:

$$\begin{aligned} T_{ID,P0,SND} &= T_{ID,P0\text{-}P1,REF} + S_{ID,P0} \\ &= T_{ID} + \Delta_{ID,P1,RCVWND}|_{min} + W_{ID,Deskew,pre} - R_{PP} \end{aligned} \tag{G.32}$$

And:

$$\begin{aligned} T_{ID,P1,RCV,E} &= T_{ID,P0,SND} + R_{PP} \\ &= T_{ID} + \Delta_{ID,P1,RCVWND}|_{min} + W_{ID,Deskew,pre} \end{aligned} \tag{G.33}$$

G.2.5.2. Bound on the duration of the Initial Diagnosis protocol

Let $T_{ID,P1,C}$ denote the local-time at which the Computation Process outputs the result for process P1.

$$T_{ID,P1,C} = T_{ID,P1,RCV,E} + W_{ID,Deskew,post} + C_{ID,P1} \quad \text{(G.34)}$$

Let $T_{ID,P1,END}$ denote the local-time at which the execution of the Initial Diagnosis protocol ends.

$$T_{ID,P1,END} = T_{ID,P1,C} + \Delta_{ID,P1,C\text{-}END} \quad \text{(G.35)}$$

The duration of the execution of the Initial Diagnosis protocol is:

$$\Delta_{ID} = T_{ID,P1,END} - T_{ID}$$

$$= \Delta_{ID,P1,RCVWND}|_{min} + W_{ID,Deskew} + C_{ID,P1} + \Delta_{ID,P1,C\text{-}END} \quad \text{(G.36)}$$

G.2.6. Initial Synchronization

Let T_{IS} denote the local time triggering the execution of the Initial Synchronization protocol. $\Delta_{IS,begin}$ denotes the delay from the end of Initial Diagnosis to the beginning of Initial Synchronization, measured in units of local clock ticks. The value of $\Delta_{IS,begin}$ is determined by the implementation and assumed constant.

$$\Delta_{IS,begin} = T_{IS} - T_{ID,P1,END} \quad \text{(G.37)}$$

G.2.6.1. Bound on the relative skew at the beginning of the Initial Synchronization protocol

Let $\pi_{IS,BEGIN}$ denote the upper bound on the relative local-time skew at the beginning of the Initial Synchronization protocol, measured in nominal clock ticks.

$$\pi_{IS,BEGIN} = \pi_{CIM,BEGIN} + [(1 + \rho_0) - 1/(1 + \rho_0)](\Delta_{ID,begin} + \Delta_{ID} + \Delta_{IS,begin}) \quad \text{(G.38)}$$

G.2.6.2. Communication between processes P0 and P1

This is discussed in Appendix C. There, π_{IS} denotes the bound on the relative skew during the execution of the Initial Synchronization protocol. Thus:

$$\pi_{IS} = \pi_{ID+IS} \quad \text{(G.39)}$$

G.2.6.3. Bound on the duration of the Initial Synchronization protocol

$\delta_{IS}|_{max}$ denotes the upper bound on the real-time duration of the execution of the Initial Synchronization protocol measured from the earliest time at which a node begins executing the protocol to the latest time at which a node applies the synchronization reset. $\delta_{IS,sync}|_{max}$, $\Delta_{IS,P2,H,h}$, $\Delta_{IS,P3,H,h}$, and $\Delta_{IS,P4,H,h}$ are given by $\delta_{sync}|_{max}$, $\Delta_{sync,P2,H,h}$, $\Delta_{sync,P3,H,h}$, and $\Delta_{sync,P4,H,h}$ in the Appendix C with B_{P0} replaced by $B_{IS,P0}$.

$$\delta_{IS}|_{max} = \pi_{IS,BEGIN} + \max(\Delta_{IS,P2,H,h}, \Delta_{IS,P3,H,h}, \Delta_{IS,P4,H,h}) \quad \text{(G.40)}$$

Let $\Delta_{IS}|_{max}$ denotes the upper bound on the duration of the execution of the Initial Synchronization protocol measured in local clock ticks. We want the fastest count of $\Delta_{IS}|_{max}$ ticks to be larger than $\delta_{IS}|_{max}$.

$$\Delta_{IS}|_{max}/(1 + \rho_0) \geq \delta_{IS}|_{max} \quad \text{(G.41)}$$

We choose the following value for $\Delta_{IS}|_{max}$.

$$\Delta_{IS}|_{max} = \lceil (1 + \rho_0)\delta_{IS}|_{max} \rceil \quad \text{(G.42)}$$

G.2.7. Bound on the relative skew during Initial Diagnosis and Initial Synchronization

The bound on the relative local-time skew during Initial Diagnosis and Initial Synchronization is:

$$\pi_{ID+IS} = \pi_{IS,BEGIN} + [(1 + \rho_0) - 1/(1 + \rho_0)]\delta_{IS}|_{max} \quad \text{(G.43)}$$

The following variables are defined in order to simplify the expressions presented below.

$$X_{IS,H,h} = \max(\Delta_{IS,P2,H,h}, \Delta_{IS,P3,H,h}, \Delta_{IS,P4,H,h}) \quad \text{(G.44)}$$

$$\sigma_0 = [(1 + \rho_0) - 1/(1 + \rho_0)] \quad \text{(G.45)}$$

Then:

$$\begin{aligned}\pi_{ID+IS} &= \pi_{IS,BEGIN} + \sigma_0(\pi_{IS,BEGIN} + X_{IS,H,h}) \\ &= \sigma_0 X_{IS,H,h} + (1 + \sigma_0)\pi_{IS,BEGIN} \\ &= \sigma_0 X_{IS,H,h} + (1 + \sigma_0)[\pi_{CIM,BEGIN} + \sigma_0(\Delta_{ID,begin} + \Delta_{ID} + \Delta_{IS,begin})] \\ &= \sigma_0 X_{IS,H,h} + (1 + \sigma_0)[\pi_{CIM,BEGIN} + \sigma_0(\Delta_{ID,begin} + \Delta_{IS,begin})] + \sigma_0(1 + \sigma_0)\Delta_{ID}\end{aligned} \quad \text{(G.46)}$$

The following inequality holds for $W_{ID,Deskew}$:

$$W_{ID,Deskew} \leq 2(1 + \rho_0)[\pi_{ID+IS} + \max(\mu_{PP,l}, \mu_{PP,h})] + 1 \quad \text{(G.47)}$$

Applying this inequality to Δ_{ID}, then:

$$\begin{aligned}\pi_{ID+IS} \leq\ & \sigma_0 X_{IS,H,h} + (1 + \sigma_0)[\pi_{CIM,BEGIN} + \sigma_0(\Delta_{ID,begin} + \Delta_{IS,begin})] \\ & + \sigma_0(1 + \sigma_0)\{\Delta_{ID,P1,RCVWND}|_{min} + C_{ID,P1} + \Delta_{ID,P1,C\text{-}END} \\ & + [2(1 + \rho_0)(\pi_{ID+IS} + \max(\mu_{PP,l}, \mu_{PP,h})) + 1]\}\end{aligned} \quad \text{(G.48)}$$

Again, the definition of the following variable simplifies the presentation.

$$Y = \sigma_0 X_{IS,H,h} + (1 + \sigma_0)[\pi_{CIM,BEGIN} + \sigma_0(\Delta_{ID,begin} + \Delta_{IS,begin})]$$

$$+ \sigma_0(1 + \sigma_0)(\Delta_{ID,P1,RCVWND}|_{min} + C_{ID,P1} + \Delta_{ID,P1,C\text{-}END}) \quad \text{(G.49)}$$

So:

$$\pi_{ID+IS} \leq Y + \sigma_0(1 + \sigma_0)[2(1 + \rho_0)(\pi_{ID+IS} + \max(\mu_{PP,l}, \mu_{PP,h})) + 1]$$

$$\pi_{ID+IS} \leq \{Y + \sigma_0(1 + \sigma_0)[2(1 + \rho_0)\max(\mu_{PP,l}, \mu_{PP,h}) + 1]\}/\{1 - 2\sigma_0(1 + \sigma_0)(1 + \rho_0)\} \quad \text{(G.50)}$$

We choose the right side of this expression as the value for π_{ID+IS}.

$$\pi_{ID+IS} = \{Y + \sigma_0(1 + \sigma_0)[2(1 + \rho_0)\max(\mu_{PP,l}, \mu_{PP,h}) + 1]\}/\{1 - 2\sigma_0(1 + \sigma_0)(1 + \rho_0)\} \quad \text{(G.51)}$$

G.3. Clique join

The clique-join recovery cases include joining after a power-on enable trigger and rejoining triggered by the detection of a failure. A clique is present on the bus for these recovery cases, and thus the major mode path through the Clique Join mode is followed to reach the Clique Preservation mode. The most important element of clique-join recovery is the loading of the state information from the clique.

The full state of the clique consists of the local time, the diagnostic state (i.e., suspicions, accusations, and convictions), and the PE communication schedule. All of these state variables are recomputed in each execution cycle. The local time is recomputed periodically by the Synchronization Preservation protocol. The suspicions are accumulated during a diagnostic cycle and then cleared after being processed to generate the suspicions-based accusations. The accusations are also accumulated during a diagnostic cycle and cleared when the convictions are updated. The convictions are recomputed at the end of every diagnostic cycle by the Collective Diagnosis protocol. The PE communication schedule is loaded anew in the Schedule Update mode immediately after each execution of the Collective Diagnosis protocol.

This state-update pattern results in a straightforward state-acquisition process for a recovering node. Because the suspicions and accusations are cleared every diagnostic cycle, keeping up with a clique after synchronizing and loading the convictions is a matter of receiving and processing messages as the clique members do. The most critical aspect of the recovery process is achieving the proper diagnostic state before attempting to capture the time and convictions state variables. To ensure a successful state loading, the sets of eligible voters in Synchronization Acquisition and Collective Diagnosis Acquisition should have more untrustworthy nodes that untrustworthy ones. The probability of achieving this is limited by the ability of Local Diagnosis Acquisition to correctly diagnose the observed nodes.

[Pike 05] presents a formal verification of synchronization sequence, including the protocols in the Local Diagnosis Acquisition and Synchronization Acquisition modes.

The presentation for the clique-initialization recovery cases up to the completion of the Clique Detection mode applies for clique-join recovery. No further timing analysis is presented here.

References

[Avizienis 04] Avizienis, Algirdas; Laprie, Jean-Claude; Randell, Brian; and Landwehr, Carl: Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Transactions on Dependable and Secure Computing*, Vol. 1, No. 1, January-March 2004, pp. 11-33.

[Davies 78] Davies, Daniel; and Wakerly, John F.: Synchronization and matching in redundant systems. *IEEE Transactions on Computers*, 27(6), June 1978, pp. 531-539.

[De Micheli 94] De Micheli, Giovanni: *Synthesis and Optimization of Digital Circuits*. McGraw-Hill, 1994.

[Driscoll 03] Driscoll, Kevin; Hall, Brendan; Sivencrona, Hakan; and Zumsteg, Phil: Byzantine Fault Tolerance, from Theory to Reality. *22nd International Conference on Computer Safety, Reliability and Security (SAFECOMP03)*, Edinburgh, Scotland, UK, October 2003, pp. 235-248.

[Geser 04] Geser, Alfons; and Miner, Paul: *A New On-line Diagnosis Protocol for the SPIDER Family of Byzantine Fault Tolerant Architectures*. NASA TM-2004-212432, 2004.

[Kopetz 87] Kopetz, Hermann; and Ochsenreiter, Wilhelm: Clock Synchronization in Distributed Real-Time Systems. *IEEE Transactions on Computers*, Vol. C-36, No. 8, August 1987, pp. 933-940.

[Lala 91] Lala, Jaynarayan H.; Harper, Richard E.; and Alger, Linda S.: A Design Approach for Ultrareliable Real-Time Systems. *IEEE Computer*, Vol. 24, No.5, May 1991, pp. 12-22.

[Laprie 95] Laprie, Jean-Claude: Dependability − Its Attributes, Impairments and Means. *Predictably Dependable Computing Systems*, B. Randell, J.-C. Laprie, H. Kopetz, and B. Littlewood, eds., Springer, 1995, pp. 3-24.

[Latronico 04] Latronico, Elizabeth; Miner, Paul; and Koopman, Philip: Quantifying the Reliability of Proven SPIDER Group Membership Service Guarantees. *Dependable Systems and Networks (DSN)*, 2004.

[Miner 02] Miner, Paul S.; Malekpour, Mahyar; and Torres, Wilfredo: A Conceptual Design for a Reliable Optical Bus (ROBUS). Presented at the *21st Digital Avionics Systems Conference (DASC)*, Irvine, California, October 27-31, 2002.

[Miner 04] Miner, Paul S.; Geser, Alfons; Pike, Lee; and Maddalon, Jeffrey: A Unified Fault-Tolerance Protocol. *Formal Techniques, Modeling and Analysis of Timed and Fault-Tolerant Systems (FORMATS-FTRTFT)*, Yassine Lakhnech and Sergio Yovine, eds., volume 3253 of Lecture Notes in Computer Science, Springer, 2004, pp. 167-182.

[Pike 04] Pike, Lee; Miner, Paul; and Torres-Pomales, Wilfredo: *Model Checking Failed Conjectures in Theorem Proving: A Case Study*. NASA TM-2004-213278, 2004.

[Pike 05] Pike, Lee: *The Formal Verification of a Reintegration Protocol*. NASA TM, 2005. To be published.

[Rushby 03] Rushby, John: *A Comparison of Bus Architectures for Safety-Critical Embedded Systems*. NASA CR-2003-212161, March 2003.

[Shin 87] Shin, K; and Ramanathan, P.: Diagnosis of processors with Byzantine faults in a distributed computing system. *17th Fault Tolerant Computing Symposium (FTCS 17)*, 1987, pp. 55-60.

[Smith 84] Smith, T. Basil: Fault Tolerant Processor Concepts and Operation. *The Fourteenth International Conference on Fault-Tolerant Computing (FTCS 14)*, Kissimmee, Florida, June 20-22, 1984, pp. 158-163.

[Srikanth 87] Srikanth, T. K.; and Toueg, Sam: Optimal Clock Synchronization. *Journal of the Association for Computing Machinery*, Vol. 34, No. 3, July 1987, pp. 626-645.

[Suri 95] Suri, N, et al: *Advances in Ultra-Dependable Distributed Systems*. IEEE Computer Society Press, 1995.

[Thambidurai 88] Thambidurai, Philip; and Park, You-Keun: Interactive consistency with multiple failure modes. In *7th Reliable Distributed System Symposium*, October 1988, pp. 93-100.

[Walter 97] Walter, Chris J.; Lincoln, Patrick; and Suri, Neeraj: Formally verified on-line diagnosis. *IEEE Transactions on Software Engineering*, 23(11), November 1997, pp. 684-721.

[XAPP077] Xilinx Application Note XAPP077: *Metastability Considerations*. version 1.0, Xilinx Corporation, January 1997.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
01- 03 - 2005	Technical Memorandum	

4. TITLE AND SUBTITLE

ROBUS-2: A Fault-Tolerant Broadcast Communication System

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Torres-Pomales, Wilfredo; Malekpour, Mahyar R.; and Miner, Paul S.

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

23-063-30-RF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

NASA Langley Research Center
Hampton, VA 23681-2199

8. PERFORMING ORGANIZATION REPORT NUMBER

L-19099

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSOR/MONITOR'S ACRONYM(S)

NASA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

NASA/TM-2005-213540

12. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Category 62
Availability: NASA CASI (301) 621-0390

13. SUPPLEMENTARY NOTES

An electronic version can be found at http://ntrs.nasa.gov

14. ABSTRACT

The Reliable Optical Bus (ROBUS) is the core communication system of the Scalable Processor Independent Design for Enhanced Reliability (SPIDER), a general-purpose fault-tolerant integrated modular architecture currently under development at NASA Langley ResearchCenter. The ROBUS is a time-division multiple access (TDMA) broadcast communication system with medium access control by means of time-indexed communication schedule. ROBUS-2 is a developmental version of the ROBUS providing guaranteed fault-tolerant services to the attached processing elements (PEs), in the presence of a bounded number of faults. These services include message broadcast (Byzantine Agreement), dynamic communication schedule update, clock synchronization, and distributed diagnosis (group membership). The ROBUS also features fault-tolerant startup and restart capabilities. ROBUS-2 is tolerant to internal as well as PE faults, and incorporates a dynamic self-reconfiguration capability driven by the internal diagnostic system. This version of the ROBUS is intended for laboratory experimentation and demonstrations of the capability to reintegrate failed nodes, dynamically update the communication schedule, and tolerate and recover from correlated transient faults.

15. SUBJECT TERMS

Avionics bus; Broadcast communication; Fault tolerance; Integrated modular architecture

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			STI Help Desk (email: help@sti.nasa.gov)
					19b. TELEPHONE NUMBER *(Include area code)*
U	U	U	UU	201	(301) 621-0390