High-Coverage Fault Tolerance in Real-Time Systems Based on Point-to-Point Communication

K. H. (Kane) Kim * and Chittur Subbaraman

Dept. of Electrical & Computer Engineering
University of California
Irvine, CA, 92697, U.S.A.
kane@ece.uci.edu (*: contact author)

and

Eltefaat Shokri

SoHaR, Inc.
Beverly Hills, CA 90211,
U.S.A.
shokri@sohar.com

Abstract: The *distributed recovery block* (DRB) scheme is a widely applicable approach for realizing both hardware and software fault tolerance in real-time distributed and parallel computer systems. One of the most important extensions of the DRB scheme which were outlined in recent years but not developed fully is the integration of the DRB scheme and a network surveillance (NS) scheme. We recently developed an NS scheme effective in a variety of point-to-point networks and it is called the *supervisor-based NS* (SNS) scheme. In this paper, we present an integration of the DRB scheme with the SNS scheme, called the *DRB/SNS* scheme. This scheme is a significant improvement over the previous versions of the DRB scheme with respect to the fault coverage and recovery time bound achieved in the systems that are based on point-to-point networks. The execution support for the integrated scheme has been implemented as a part of the DREAM kernel prototype, a timeliness-guaranteed operating system kernel developed at the University of California, Irvine. The recovery time bound of the DRB/SNS scheme is analyzed on the basis of the prototype implementation.

Keywords: Distributed recovery block, network surveillance, point-to-point networks, real-time systems, fault-tolerance, fault coverage, recovery time bound.

1. Introduction

The complexity of large-scale real-time systems used in safety-critical applications has been growing rapidly in recent years [Bas96, Yen97]. Such applications impose stringent reliability requirements on the computing systems, including the requirements that recovery from both hardware and software faults be *tightly bounded in time*. The *recovery time bound* must also be acceptably small. Moreover, it is highly desirable to use fault tolerance techniques that are *scalable*, i.e., applicable to architectures of different sizes.

One broadly applicable *software approach* for realizing real-time fault tolerance capabilities in parallel and distributed computer systems is the distributed recovery block (DRB) scheme [Kim94, Kim95a, Fra91, Hec91]. The DRB scheme was created as an approach for parallel redundant execution of real-time tasks such that if hardware and/or software faults occur, forward recovery with minimal bounded delay is effected. In recent years, the DRB scheme has been extended in several directions to fit into different types of system architectures and also to further improve the recovery time bound and the fault coverage of the scheme [Kim96].

One of the most important extensions which were outlined in recent years but not developed fully is the integration of the DRB scheme and a *network surveillance* (NS) scheme. NS schemes facilitate the fast learning by each interested fault-free node in the system of the faults or the repair completion events occurring in other parts of the system and also facilitate fast reconfiguration. We recently developed a semi-centralized real-time NS scheme [Kim97] effective in a variety of point-to-point networks and it is called the supervisor-based network surveillance (SNS) scheme.

In this paper, we present an integration of the DRB scheme with the SNS scheme, called the DRB/SNS scheme. This scheme is a significant improvement over the previous versions of the DRB scheme with respect to the fault coverage and recovery time bound achieved in the systems that are based on point-to-point networks. Moreover, this scheme is one of the very few software approaches for delay-bounded tolerance of both hardware and software faults that can be used for point-to-point network architectures. An integration of the DRB scheme with a decentralized NS scheme for local area network (LAN)-based systems with efficient broadcast capabilities was outlined earlier in [Kim96]. The problems encountered in integrating the DRB approach with an NS scheme effective in a point-to-point network environment are substantially different from those in an efficient broadcasting LAN environment. We also present a concrete modular implementation model of the DRB/SNS scheme.

As a validation experiment, the execution support for the scheme was incorporated into a prototype implementation of the DREAM kernel which is a model of a timeliness-guaranteed operating system kernel developed at the University of California, Irvine [Kim95b]. This implementation minus a few low-level device-dependent modules can be viewed as an implementation model which can be easily adapted to various commercial operating system kernels.

In order to obtain some tight recovery time bounds, the performance of the DRB/SNS scheme is analyzed on

the basis of the presented implementation model under various cases of faults. In safety-critical real-time distributed computing applications, the recovery time bound is a measure of critical importance, but yet such analyses had been done scarcely until a few years ago.

The paper starts in Section 2 with a description of the point-to-point network based system architecture model adopted and also the fault source model along with the assumed failure frequency bounds. The SNS scheme developed earlier is reviewed in Section 3. Section 4 then presents the basic principles and a modular implementation model of the DRB/SNS scheme. A recovery time bound analysis of the DRB/SNS scheme is given in Section 5 and the paper concludes in Section 6.

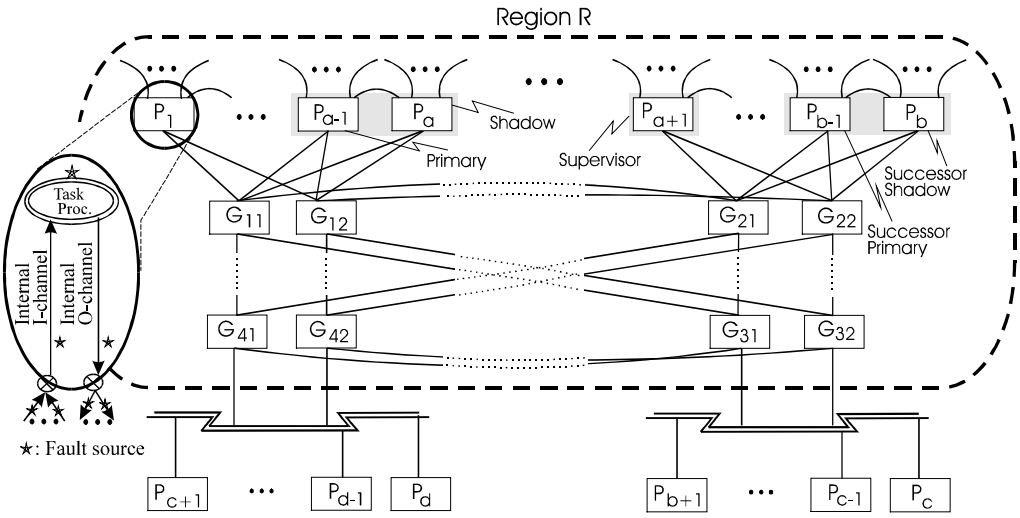

Figure 1. System architecture model

2. System architecture model, fault types, and fault frequency bounds

As a system architecture model which can represent a variety of point-to-point network architectures with different specializations, the model depicted in Figure 1 was chosen. The system consists of *m* different local area networks (LAN's). Each LAN consists of a maximum of *n* different local nodes and typically contains multiple gateway nodes. Within a LAN, the local nodes may be connected with one another either via a broadcast bus or via a point-to-point network. Two nodes in different LAN's communicate with each other through the gateways attached to each LAN. Figure 2 also shows a region *R* bounded by dotted lines. The nodes within *R* communicate with each other via one or more point-to-point links. These nodes execute the SNS scheme. Only a subset of them may be involved in executing a real-time application structured and operating under the DRB/SNS scheme. For the nodes in each LAN with efficient broadcast facilities, which are outside *R*, some alternate NS scheme such as the periodic reception history broadcast (PRHB) [Kop93, Kim94a] or the time-triggered protocol (TTP) [Kop93] which takes advantage of the cheap broadcast facility may be chosen. From now on, we concentrate on the nodes in *R* for application of the DRB/SNS scheme.

Every message sent from a source node to a non-neighbor destination node within *R* is stored for a while in each intermediate node while the node makes a routing decision. Such a routing scheme is called a *store-and-forward* routing scheme. This type of system architecture or its minor variations can be found in applications such as defense command-control systems, factory automation systems, etc. As shown in Figure 1, each LAN has two or more gateways which function in a redundant fashion. Thus, even if one of the gateways becomes faulty, each node in a LAN can still communicate with a node in a different LAN through another gateway.

Two nodes in *R* are said to be *neighbors* of each other if they are connected by a direct point-to-point link. For example P_a and G_{11} are neighbors of each other.

2.1 Fault sources

In the model depicted in Figure 1, possible sources of faults in a node are represented by a *task process*, an *incoming communication handling channel* (I-channel), and an *outgoing communication handling channel* (O-channel). *Faults in the task process* represent faults in the main processor, faults in the application software, faults in the system software running on the main processor, etc. *Faults in the I-channel* represent faults in various components of a node (both hardware and software) that are involved in receiving a message from the point-to-point network. *Faults in the O-channel* represent faults in various components of a node (both hardware and software) that are involved in sending a message to the network. Any observed fault of a node could be an instance of a combination of faults in the fault sources mentioned above.

The *remaining fault source* is one or more links of the point-to-point interconnection network. Faults in the interconnection network inside a DRB station would affect the intra-station communication whereas the faults in the interconnection network outside a DRB station may affect the inter-station communication. Each of the four fault sources described in the above fault source model is called a fault source component.

We assume that since the routing scheme is of the store-and-forward type, a permanent failure of any one of the three fault source components in a node disables not only the node's processing capabilities but also the node's routing capabilities.

2.2 Fault types covered and fault frequencies assumed

The DRB/SNS scheme has been designed to tolerate various types of faults that occur in various fault sources mentioned in the preceding section subject to the fault frequency assumptions stated below. Of the assumptions, A1-A4 have been made for the SNS scheme.

(A1) The task process and two other fault-source components in each node do not generate messages containing erroneous values nor untimely messages.
(A2) The clocks in the nodes are kept synchronized sufficiently closely for practical purposes, i.e., for the given applications.
(A3) Each of the nodes performing store-and-forward functions (as well as the source node) transmits each stored message twice continuously. It is assumed that this makes the probability of transient fault occurrences in the components of the two neighbor nodes and those in the link between the two neighbor nodes causing message losses to be negligible. This dual redundant transmission approach was also exploited in [Kop93].
(A4) Let S_N be the set of fault source components belonging to a node N, and $LL(P, Q)$ be {link connecting two neighbor nodes P and Q}. The time interval from the time of the occurrence of a permanent hardware fault F in S_N or in $LL(P, Q)$ to the time every healthy node in the system learns of the fault occurrence under the SNS scheme is the *detection period* of the SNS scheme for the fault F. The SNS scheme assumes that during that detection period, no other hardware fault occurs in node M, where $M \neq N$, or in $LL(I, J)$, where $(I, J) \neq (P, Q)$. The *worst-case detection latency of the SNS scheme* for all possible hardware faults in the system is denoted by WDL_{SNS}. Therefore, it is assumed that no second hardware fault occurs in the system within WDL_{SNS} after the occurrence of the first permanent hardware fault.

Let RT denote the maximum interval of time between the occurrence of a permanent fault in a DRB station and the completion of the repairing of the faulty situation which may involve relocating a member node of the station. It is safe to say that RT is much larger than WDL_{SNS}.
(A5) In a time interval of RT, the permanent failures of both partner nodes in a DRB station are assumed to have a negligible probability of occurrence.
(A6) In a time interval of RT, the transient failures of both partner nodes in a DRB station are assumed to have a negligible probability of occurrence.
(A7) The topology of the point-to-point network used, the nature of the application, and the task relocation arrangement are such that the node and link failures occurring within the bounds of A4, A5, and A6 cannot lead to permanent partitioning of the application system into inoperable disconnected subsystems during the lifetime of the application mission.

Under the DRB/SNS scheme, two copies of every message are sent from a source node to a destination node, the first copy along one path, P_1 and the second copy along an alternate path, P_2. It is assumed that it is always possible to find P_1 and P_2 such that they have no links/nodes in common, except the source node and the destination node.

If this assumption A7 along with A3 and A4 hold, it implies that a message sent from the source node will definitely reach the destination node as long as the destination node does not fail during the message transmission period. (This is because during the message transmission period (which is much smaller than WDL_{SNS}) no two permanent faults can occur and thus at least one path, P_1 or P_2, is fault-free. Also, transient faults are masked as stated in A3.)

3. Overview of the SNS scheme

Figure 2 shows the basic operation of the SNS scheme. As shown in the figure, there are two types of nodes that execute the SNS scheme, the *worker* nodes and a *supervisor* node. The worker nodes are mainly responsible for judging their own health status, the health status of their neighbor nodes, and the health status of the links attached to themselves. The supervisor node performs all the duties that a worker normally does.

In addition, it is responsible for collecting *fault suspicion reports* from worker nodes, using the collected information to judge whether a fault has indeed occurred, and then sending the *fault occurrence notice* to all the healthy worker nodes in the system.

The SNS scheme is executed by a set of nodes S_{NSR} that consists of n worker nodes and a supervisor node. Under the normal mode of operation, each node in S_{NSR} will have at least two healthy neighboring nodes. If the number of neighbors of a node N in S_{NSR} drops down to one, then we treat N as a component of its neighbor node Y. Thereafter, Y and its neighbors are responsible for detecting the faults in N. A subset of S_{NSR} is involved in actively executing some distributed real-time application tasks. In Figure 1, S_{NSR} would be all the nodes within the region R or a subset of them.

Under the SNS scheme, *two copies* of every message are sent from a source node to a destination node, the first copy along one path P_1 and the second copy along a disjoint alternate path P_2. Since it is assumed that (a) transient faults are masked by the dual redundant transmission approach (assumption A3), (b) permanent faults cannot occur in more than one node or link during the period of a message transmission which is much shorter than WDL_{SNS} (assumption A4), and (c) node or link failures do not lead to permanent partitioning of the application system during the lifetime of the application mission (assumption A7), it is always possible for the

source node to find P_1 and P_2 such that at least one copy of the message sent will definitely reach the destination node as long as the source node and the destination node are healthy during the period of the message transmission.

4. The DRB/SNS Scheme

4.1 Basic principles of the DRB scheme

The distributed recovery block (DRB) scheme is an approach for realizing both hardware fault tolerance and software fault tolerance in real-time distributed and/or parallel computer systems. In its basic configuration, a *DRB computing station* consists of two processing nodes executing two functionally equivalent tasks in parallel, the first node called the primary node and the second node called the shadow node, where a computing station refers to a processing node (hardware and software) dedicated to the execution of one or a few application tasks [Kim94, Kim95a]. Here the task software in the primary node as well as that in the shadow node are constructed by use of the recovery block language construct [Ran95]. A recovery block may consist of multiple versions of a task procedure and an acceptance test (AT) function designed to judge the reasonableness of the results produced by each version. Such versions are called try blocks. A try (i.e., execution of a try block) is always followed by an AT execution. A try not completed within the maximum execution time allowed for each try block due to hardware faults or excessive looping is also treated as an AT failure. Therefore, the AT is a combination of both *logic* and *time* AT's and can be constructed entirely in software or in the form of a software-hardware combination. In most cases a recovery block containing just two try blocks, a *primary try block* and an *alternate try block*, is designed. The roles of the two try blocks are assigned differently in the two partner nodes. The governing rule is that *the primary node tries to execute the primary try block whenever possible whereas the shadow node tries to execute the alternate try block.*

4.2 Operating rules of the DRB/SNS scheme

In typical real-time fault-tolerant computer systems using the DRB scheme, there will be several DRB stations among which real-time data flow. These stations are called here worker DRB stations. In addition, we may also incorporate a supervisor station in order to make the system highly robust and extend the lifetime of the system [Hec91, Kim95a] as shown in Figure 1. Actually, the SNS scheme already incorporates a supervisor node that monitors the health status of all the worker nodes as well as the inter-node communication links in the system, and in case of a fault observation, informs all the healthy nodes in the system of the observed fault. In the DRB/SNS scheme, the supervisor node performs all these functions and in addition, is responsible for detection of

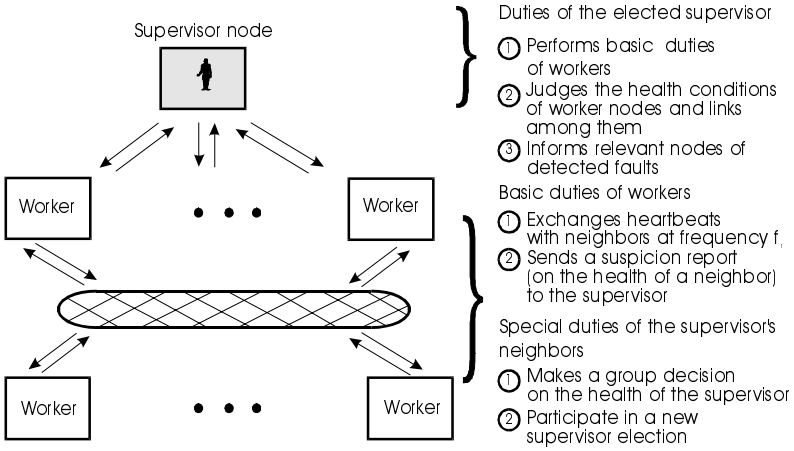

Figure 2. The SNS scheme

the misjudgments made by the nodes in DRB stations about the status of their partner nodes, reconfiguration of the network including task redistribution, etc.

The processing in a worker DRB station begins when the worker DRB nodes receive an input data item from the predecessor DRB station. Next, the task process in the primary node picks a data item for processing and sends the ID of the data item to its partner (event O1). This step is necessary to ensure consistent processing of input data by the partner worker nodes in the DRB station. Also, if the data ID does not arrive at the shadow node within a specific deadline DL_{ID}, the shadow knows that a fault has occurred somewhere. In such a case, the shadow task process will change its role to that of the primary, pick up a new input data item to process and execute its try block and AT. Upon passing the AT and before updating its local data base and sending the output message out, the new primary will seek the help of the SNS scheme executing within the primary node to find out if any permanent I-channel fault, in short a PI fault, has occurred within the primary node and caused the node to miss the data ID message from the partner node and switch its role to change from that of the shadow to that of the primary. For this, the new primary checks whether WDL_{SNS} time units have gone past since the time the data ID message would have arrived at the primary node under normal fault-free circumstances. This step is necessary to prevent the two partner nodes from entering dangerous inconsistent states. Only after the new primary task process confirms that the primary node has been fault-free, it will produce the output. Otherwise, it will initiate a spontaneous shutdown. Thus, a faulty node does not produce any external outputs.

After the O1 event, the primary task process proceeds to execute its primary try block and test the validity of the result using the AT. Upon completing the AT execution, it sends the AT result (ATR) to the partner (event O2). This message (or absence of it within a specific deadline, DL_{AT}) helps the partner in detecting faults (software or hardware) in the primary node. After

the O2 event, the primary task process proceeds to send the output to both its successor primary and the successor shadow nodes (event O3). Since this output is sent in a redundant fashion (similar to the case discussed in Section 3) to each of the two successor nodes (primary and shadow), the message is bound to arrive at the receiver nodes as long as the sender and the receiver nodes remain fault-free. Hence, the primary node need not wait or take an extra action to confirm the successful delivery of the output message to the successor DRB station but rather can assume that the message it sends out will reach at least one of the nodes of the successor DRB station which will remain fault-free during the transmission period. Thus, the primary node sends the output success notice (OSN) to its partner node (event O4) after sending the output to the successor DRB station. This message (or absence of it within a specific deadline, DL_{OSN}) helps the partner in detecting faults in the primary node.

4.3 Modular structuring of the DRB/SNS scheme

Figure 3 shows a modular implementation model for the primary node. The structuring of the shadow node is similar and will not be discussed in this paper. The function of the primary DRB node consists of the *Application Task Process* and four co-resident kernel-threads called the *Watchdog Timer Thread* (WTT), the *Incoming Communication Thread* (ICT), the *Outgoing Communication Thread* (OCT), and the *Network Surveillance & Partnership Support Thread* (NPT). The four kernel-threads are periodic threads with the unused portions of thread-time-slices "donated" to the task process. Inter-process communication (IPC) within a node is achieved through various queues located in the shared memory area of the node whereas IPC between two different nodes is achieved by message transmission through the point-to-point interconnection network.

The functions of the application task process and the four kernel-threads are as follows:

(1) **Application task process**: This is the main process that performs the application computation. This process consists of the primary logic of the application task and the mechanisms to support the DRB/SNS scheme.
(2) **WTT**: This kernel-thread checks whether the application task process violates the try block execution deadlines set in $Q_{deadline}$, and in case a deadline violation is detected, it orders the task process to turn to a shadow task process through a message deposited in $Q_{timeout}$.
(3) **NPT**: This is responsible for almost everything in the SNS scheme described in Section 3 except the generation of the heartbeat signals. For example, some or all of the functions such as periodic analysis of the received heartbeat signals, reporting fault suspicions, election of a supervisor, and supervisory function, are handled by this kernel-thread, depending on the role of the host node (a worker node, supervisor node or a neighbor of the supervisor node). This thread also notifies the application task process of any detected faults in the host node (by depositing an order to shut down in Q_{order} checked by the application task process). In addition, this thread honors any requests from the partner or supervisor. For instance, the partner may request for a copy of a missing data item, the supervisor may order the node to shut the latter's operation down, etc.
(4) **ICT**: This distributes the messages received from other nodes to appropriate destination queues. In particular, it periodically forwards the messages that should be processed by NPT into Q_{NPT} and the input data items into Q_{IN} checked by the application task process. In addition, it also maintains a buffer containing some number of the most recent input data items. This buffer is used by NPT to honor a partner's request for a particular missing input data item.
(5) **OCT**: This sends to the point-to-point network messages such as output data items deposited into Q_{OUT} by the application task process and the messages related to the SNS scheme deposited into Q_{OUT} by NPT. Also, this generates heartbeat signals to be sent to each of the node's healthy neighbors as a part of the SNS scheme.

If any fault occurs in the primary node, the shadow task process will learn of the fault either by an explicit notice from the primary (as in the case of an AT failure in the primary) or in one of the following four ways:
(a) a notice from the WTT of the shadow node regarding the absence of a data ID from the primary partner within a specific deadline DL_{ID}, or
(b) a notice from the WTT of the shadow node regarding the absence of an AT result output from the primary partner within a specific deadline DL_{AT}, or
(c) a notice from the WTT of the shadow node regarding the absence of an OSN message from the primary partner within a specific deadline DL_{OSN}, or
(d) a notice from the NPT of the shadow node that the primary node has failed.

Although all the detailed logical operations of the DRB/SNS scheme are embedded within the kernel in the implementation model in Figure 3, it is possible to implement middleware components performing essentially the same logical operations and running on top of many commercial real-time kernels.

5. Recovery time bound analysis of the DRB/SNS scheme

Notations

(N1) **Time *t(a)***: Time *t(a)* is defined as the instant at which an event *a* occurs.
(N2) **Time interval *d(a,b)***: Let *a* and *b* denote two types of ordered events that take place in the order [*a, b*]. Then *d(a,b)* is defined as the time interval between the start of *a* and the end of *b*. ■

5.1 Definitions

(1) **Processing time *L* of a DRB station**: *L* is defined as the time interval between the time of arrival of a particular

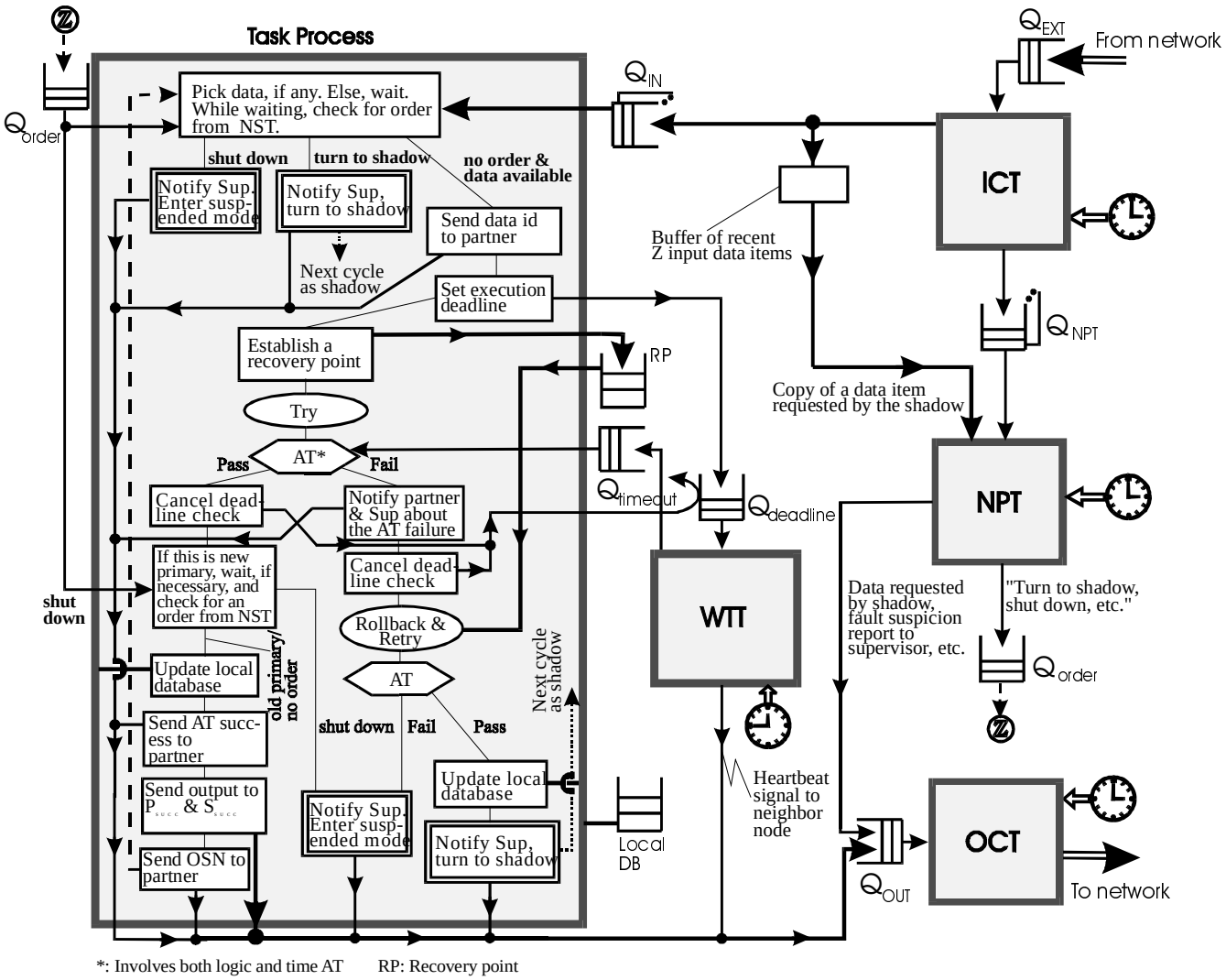

Figure 3. Modular implementation model of the DRB/SNS scheme

data item at Q_{EXT} of the primary node and the time the corresponding output leaves the DRB station.

(2) **Worst-case processing time L_{ff} of a DRB station in the absence of faults**: L_{ff} is defined as the maximum value of L when a DRB station operates under fault-free conditions.

(3) **Worst-case processing time L_{max} of a DRB station in the presence of faults**: L_{max} is defined as the maximum value of L when faults occur in the DRB station within the bounds established in Section 3.

(4) **Recovery time bound *RTB* of a DRB station**: *RTB* is defined as the difference between L_{max} and L_{ff}.

(5) **Maximum message transmission time *TRANS***: Maximum time interval that elapses from the time a message leaves the Q_{OUT} of a node to the time the message reaches the Q_{EXT} of any other node in the system. This time includes the time taken for redundant transmissions of the same message over two different paths as explained in Section 3.2.

(6) **Maximum thread turnaround time**: Maximum interval of time that elapses from the time an item arrives at any of the input queues of the thread to the time the corresponding output is produced.

(6a) **Maximum ICT turnaround time *MIT***: Maximum amount of time that elapses between the time of arrival of a message in the input queue of ICT in a node (i.e., Q_{EXT} in Figure 3) to the time at which the ICT completes the forwarding of the item.

(6b) **Maximum OCT turnaround time *MOT***: Maximum amount of time that elapses from the time of arrival of an item at the input queue of OCT (i.e., Q_{OUT} in Figure 5) to the time at which the OCT completes the processing (sending) of the item.

(7) **Worst-case fault detection latency WDL_{SNS} of the SNS scheme**: Maximum amount of time between the occurrence of a fault $F \in$ {node fault, link fault} and the learning of the fault occurrence by all the healthy nodes in the system. The details on a method for calculation of WDL_{SNS} are referred to [Kim97].

(8) **Lag limit *LGL* of the shadow**: This is defined as the maximum interval of time the shadow task process can lag behind the primary task process without inducing the risk of missing the deadline for taking the final output action(s). If the shadow exceeds this lag limit, it should enter a *node recovery mode* and try to catch up with the primary. ■

5.2 Recovery time bound of the DRB/SNS scheme

Figure 4 shows the timing chart of a DRB station under fault-free conditions. The three parameters of interest in Figure 4 are *MIT*, *MOT*, and *TRANS*. According to the definitions given in Section 5.1, the worst-case time interval between the arrival of an input data message in Q_{EXT} of the primary node (event $I1_p$-a) and the time at which the ICT finishes moving this message to Q_{IN} (event $I2_p$-a) in Figure 4 is shown as *MIT* time units. Similarly, the worst-case time interval between the arrival of a message in Q_{OUT} (event O1-i) and the time at which the OCT finishes sending this message to the network (event O1-s) is shown as *MOT* time units. Finally, the worst-case time interval between the event O1-s of the primary node and I2-a of the shadow node is shown as *TRANS* time units. For the simplicity in the analysis of the recovery time bounds, we assume that the difference between $t(I1_p\text{-}a)$ and $t(I1_S\text{-}a)$ is negligible. Note that L_{ff} includes the time period during which the input data item waits in Q_{IN} to be picked up by the primary task process. The bound for this waiting time should also be known at the design time.

This timing chart facilitates the calculation of various deadlines used to detect failures. The deadline in the shadow task process for the checking of the data ID from the primary for a particular input data item X is calculated as

$$\begin{aligned} &t(I1_p\text{-a}) + d(I1_p\text{-}a, O1\text{-i}) + d(O1\text{-}i, O1\text{-}s) + \\ &\quad d(O1\text{-}s, I2\text{-}a) + d(I2\text{-}a, I2\text{-}p) + LGL \\ &\quad \le t(I1_S\text{-}a) + d(I1_p\text{-}a, O1\text{-}i) + MOT + TRANS + \\ &\qquad MIT + LGL = DL_{ID} \end{aligned} \quad (1)$$

Here *LGL*, the lag limit, is the tolerable delay that can be incurred by the shadow task process in checking the arrival of the data ID from the primary partner. $t(I1_S\text{-}a)$ is the time at which X arrived at the shadow node, and $d(I1_p\text{-}a, O1\text{-}i)$ is the sum of the queuing delay of X at the primary node and the time taken by the primary task process to execute the step for sending the ID of X to the shadow. The worst-case value for $d(I1_p\text{-}a, O1\text{-}i)$, *WQD*, can be quite accurately estimated by the shadow node by using the times at which the OSN arrived at the shadow node in the immediately preceding cycle, and the time at which X arrived at the shadow node. Note that a copy of X is assumed to have arrived at both the primary and the shadow nodes roughly at the same time.

Thus, once an input data item arrives at the shadow node, the shadow task process should check for the arrival of the ID of the data item from the partner within *WQD* + *MOT* + *TRANS* + *MIT* + *LGL* time units. Suppose this "intermediate" deadline is violated. The shadow will change its role to that of the primary, pick up a new input data item, deposit the data ID in Q_{OUT}, and start the processing of a new data item. After the new primary passes the AT, it first checks whether the time interval of WDL_{SNS} units has gone past since the time at which the data ID message would have arrived at the shadow task process's data ID queue, Q_{dataid}. Note that the time at which the shadow task process checks this condition will be in the worst-case equal to $DL_{ID} + P_{exec}$, where P_{exec} is the worst-case time for executing the step of picking the new input data item, depositing the data ID in Q_{OUT}, and finally executing both the primary try block and the AT after sending the data ID (Figure 4). The time at which the data ID would have arrived at Q_{dataid} will be equal to $(DL_{ID} - LGL)$ and the shadow task process checks for the presence of the data ID *LGL* time units later in the worst-case. This means that the shadow task process waits for $Z((DL_{ID} - LGL + WDL_{SNS}) - (DL_{ID} + P_{exec})) = Z(WDL_{SNS} - (LGL + P_{exec}))$ time units after executing its try block and AT, where $Z(x) = x$ if $x > 0$ and $Z(x) = 0$ if $x \le 0$.

The worst-case processing time of the DRB station in the normal fault-free case is thus

$$L_{ff} = WQD + P_{exec} + DATR\_OM + MOT \quad (2)$$

where *DATR_OM* is the time taken to execute the steps of depositing the AT result and the output message in Q_{OUT}. As mentioned before, the shadow task process can learn of the faults occurred in the primary node in four major ways besides relying on a notice from the primary itself. The recovery time bound of the DRB/SNS scheme can be determined by considering these cases. Due to space limit, details of the analyses of the four cases are not given here and only the analyses results are given below.

Case 1: WTT in the shadow node detects the absence of a data ID from the primary by DL_{ID}

The recovery time is *MOT* + *TRANS* + *MIT* + *LGL* + $Z(WDL_{SNS} - LGL - P_{exec})$ time units.

Case 2: WTT in the shadow node detects the absence of AT result from the primary by DL_{AT}

The recovery time is *MOT* + *TRANS* + *MIT* + *LGL* time units.

Case 3: WTT in the shadow node detects the absence of OSN from the primary within DL_{OSN}

This case is similar to case 2, and the recovery time here is again *MOT* + *TRANS* + *MIT* + *LGL* time units.

Case 4: NPT in the shadow node detects that the primary node has failed.

The recovery time cannot be greater than that in the three preceding cases (1, 2, and 3).

The results of the above analysis can be summarized as follows.

Proposition P1: If a fault occurs in a model component of a DRB station operating under the DRB/SNS scheme, then the recovery time for a real-time application in the station is bounded by *RTB* = *MOT* + *TRANS* + *MIT* + *LGL*

+ Z($WDL_{SNS} - LGL - P_{exec}$), where all the parameters have the meanings given in Section 6. ■

Numerical example: In the DREAM kernel prototype, $MIT = MOT = 8$ msec. Also, in [Kim97] WDL_{SNS} for the DREAM kernel has been calculated as 95.5 msec. If we choose $TRANS = 0.5$ msec, and $LGL = 1$ msec, $P_{exec} = 200$ msec, then RTB = 17.5 msec.

6. Conclusion

In this paper, we extended the basic DRB scheme into the DRB/SNS scheme for time-bounded fault tolerance in point-to-point network architectures and the main extension was in incorporating a supervisor-based network surveillance scheme. The new extension yields better fault coverage and better recovery time bounds. A prototype implementation of the scheme has been incorporated into a DREAM kernel prototype running on a PC network. Through experimental injection of faults, the capability of the implementation was validated to some extent. An analysis procedure for recovery time bounds, which could be of major importance in many hard-real-time applications, was also presented. Some fruitful directions for future research in this area would be to integrate the DRB/SNS scheme with the scheme for configuring a new shadow node as well as to extend the DRB/SNS scheme to allow hosting more than one application task on a node.

Acknowledgments: The research work reported here was supported in part by the US Defense Advanced Research Project Agency under Contract N66001-97-C-8516 monitored by NRaD, in part by the University of California's MICRO Program under Grant 96-169, in part by Hitachi, Ltd., in part by LG Electronics, and in part by the USAF Rome Lab under Contract F30602-96-C-0060.

References

[Bas96] Bastani, F., et al., "Towards Dependable Safety-Critical Software", Proc. 2nd IEEE CS Workshop on Object-Oriented Real-Time Dependable Systems, Laguna Beach, CA, February 1996, pp. 86-92.

[Fra91] Fraga, J.S., Rodrigues, V. and Silva, E.S., "A Language Approach to the Implementation of the Distributed Recovery Block Schemes", Proc. 13th CBC Conf. on Comp. Sciences, Brazil, August 1991.

[Hec91] Hecht, M., et al., "A Distributed Fault Tolerant Architecture for Nuclear Reactor and Other Critical Process Control Applications.", Proc. IEEE CS 21st Int'l Symp. on Fault-Tolerant Computing, June 1991, Montreal, pp.462-469.

[Kim94] Kim, K.H., "Action-level Fault Tolerance", Ch. 17 in Sang H. Son ed., 'Advances in Real-Time Systems', Prentice Hall, 1994, pp. 415-434.

[Kim95a] Kim, K.H., "The Distributed Recovery Block Scheme", Ch. 8 in Michael R. Lyu, ed., 'Software Fault Tolerance', 1995, pp. 189-209.

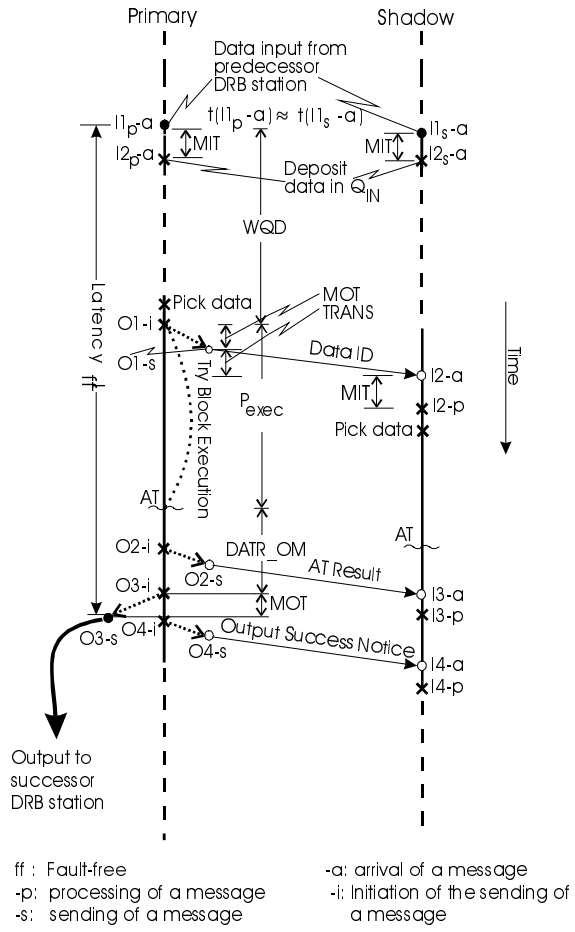

Figure 4. Timing chart of a DRB station

[Kim95b] Kim, K.H. et.al, "A Timeliness-Guaranteed Kernel Model - DREAM Kernel and Implementation Techniques", Proc. 1995 Int'l Workshop on Real-Time Computing Systems and Applications (RTCSA 95), Tokyo, Japan, Oct. 1995, pp. 80-87.

[Kim96] Kim, K.H., Bacellar, L., and Subbaraman, C., "Primary-Shadow Consistency Issues in the DRB Scheme and the Recovery Time Bounds", Proc. Int'l Symposium on Software Reliability Engineering, White Plains, NY, November 1996, pp. 319-329.

[Kim97] Kim, K.H., and Subbaraman, C., "A Supervisor-Based Semi-Centralized Network Surveillance Scheme and the Fault Detection Latency Bound", to appear in Proc. 16th Symp. on Reliable Dist. Sys., October 1997.

[Kop93] Kopetz, H. and Grunsteidl, G., "TTP-A: Time Triggered Protocol for Fault Tolerant Real-Time Systems", *Proc. IEEE CS FTCS-23*, Toulose, France, June 1993, pp. 524-533.

[Ran95] Randell, B., "The Evolution of the Recovery Block Concept", Chap. 2 in Software Fault Tolerance, Michael R. Lyu, Editor, 1995, pp. 1-21.

[Yen97] Yen, I-L., "An Object-Oriented SNMR Framework for Dependable Systems", Proc. 3rd IEEE CS Workshop on Object-Oriented Real-Time Dependable Systems, Newport Beach, CA, Feb. 1997, pp. 291-297.