A Layered Approach to Fault-Tolerance and Timeliness Issues in SAFENET

Kang G. Shin, Lup-Houh Ng
Real-Time Computing Laboratory
EECS Department
The University of Michigan
Ann Arbor, MI 48109-2122
Email: {kgshin,luphouh}@eecs.umich.edu

Timothy P. Monaghan
Code 5051
Naval Air Warfare Center
Aircraft Division
Warminster, PA 18974-0591
monaghan@nadc.navy.mil

Abstract

In this paper, we investigate the fault-tolerance issue of networking and propose a layered solution to it. By evaluating the performance of fault-handling schemes at each layer of a protocol stack, we can find a set of fault-handling schemes whose combined performance will guarantee the required end-to-end delay bound for real-time communication services while maintaining a certain level of fault-tolerance. The Survivable Adaptable Fiber Embedded Network (SAFENET) is used as the main vehicle of this study.

1 Introduction

Tasks executing in a distributed computing system often need to communicate with one another in order to accomplish a common goal. Even if two communicating tasks may be executing concurrently on two different hosts, one task may block while waiting for a message from the other. The communication delay between the two tasks will therefore have an impact on the task completion times. In time-critical applications, such as flight control and C^3I, the executing tasks must be completed before their deadlines, or a *dynamic system failure* is said to have occurred. In order to guarantee predictable system performance, the delay between two communicating real-time tasks must be bounded.

Early work on real-time communication focused on admission control, scheduling policies and schedulability conditions to guarantee end-to-end delay bounds [1–5]. However, until recently these guarantees were made under the assumption that no errors occur during the communication. These assumptions do not hold in the real world as a data packet may be corrupted during its transmission due to transient faults like electrostatic noise and other external disruptions. A message that is delivered on time but contains incorrect data is as useless, or even as disastrous, as a message that contains the correct data but missed its deadline. Thus, reliability becomes an important issue in real-time communication and it is the subject of this paper.

The work reported in this paper was supported in part by the US Naval Air Warfare Center, Warminster, PA, and the Office of Naval Research under Grant N00014-94-1-0229. Any opinions, findings, and conclusions or recommendations are those of the authors and do not necessarily reflect the views of the US Navy.

Most, if not all, of the network protocol stacks successfully implemented already have some form of fault-handling capabilities built into the protocols to provide a "reliable" network service where possible, albeit most of the protocols provide such reliability on a best-effort basis. Each of these fault-handling schemes requires certain resources like time and buffer. To determine if a real-time channel established in the protocol stack is able to meet its end-to-end delay requirement, we have to determine the performance of each fault-handling scheme involved at each layer of the protocol stack, and compute the aggregate performance. On the other hand, once we have a handle on the performance of the various fault-handling schemes at each layer of the protocol stack, we can select a set of such schemes so that the aggregate performance may meet the given end-to-end delay requirement.

The rest of this paper is organized as follows. Section 2 describes our overall approach, whereas Section 3 discusses and defines the various quantitative performance measures on which we will evaluate the fault-handling schemes. Section 4 presents an overview of the fault-handling features in SAFENET and evaluates the fault-handling schemes in the SAFENET LAN and Transport Services layers. These will be followed by a description, in Section 5, of a method for computing the combined performance measure of a fault-handling strategy and then determining if the combined performance measures meet the service requirements. We also present the simulation results there. Section 6 concludes the paper and suggests future directions.

2 The Proposed Approach

Imagine a real-time communication channel without any error-handling capability. Two tasks that are communicating through such a channel may exercise some form

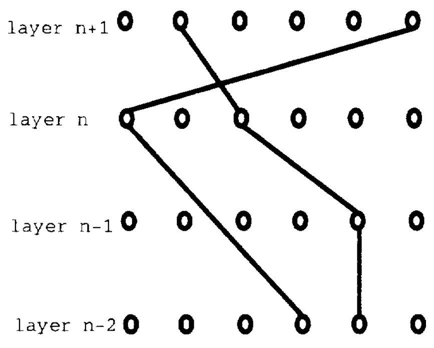


Figure 1: Selection of a fault-handling strategy

of reliable communication by checking the integrity of the messages each has received from the other in order to detect every conceivable error that could occur during transmission. This is obviously a very naive scheme since we know that certain errors could be handled more efficiently by intermediate network components during transmission while other errors may not be easily detected at the receiving end. By attempting to correct an error at the destination end of the channel, we are also allowing a large error latency that would cost the receiving task dearly in terms of time, which is the most precious resource for time-critical applications.

In fact, there are already many schemes to handle transmission errors in intermediate network components. These schemes include error correcting codes, checksumming, congestion/flow control at the network level, and so on [6,7]. However, each of these schemes may cover different types of faults and consume different system resources like transmission and processing bandwidths. By taking a layered view of the communication channel (e.g., the OSI Model), we can see each layer of the channel adopting various schemes to handle errors detectable at that layer, or avoiding faults that could occur at that layer. We shall call these *fault-handling schemes* and a selected set of these schemes a *fault-handling strategy* of the channel.

The applications that are utilizing a channel may have certain *client service requirements* like throughput, delay, and error coverage, among others. Then, a selected set of fault-handling schemes to make up the fault-handling strategy of the channel must have a combined performance measure that meets the service requirements. For example, the combined latency of the fault-handling strategy must be within the delay bound of a time-critical channel at that layer.

A different fault-handling strategy may be necessary for different channels to meet the different service requirements. If we could quantify the performance measures and the resource requirements of each fault-handling scheme at each layer, it is then possible to select an 'optimal' fault-handling strategy to meet the service requirements of the corresponding channel.

Based on the above premise, we will first investigate the performance measures of a fault-handling scheme to establish a basis for evaluation. We will then proceed to evaluate the performance of various fault-handling schemes at each layer. The results of our evaluation will be used to compute the combined performance measures of a fault-handling strategy. Specifically, we will take the following steps.

S1. Determine the performance measures of each fault-handling scheme including

- fault coverage;
- latency (processing time overhead);
- throughput (bandwidth);
- cost function, which may include factors like buffer requirements, hardware redundancy and security considerations.

S2. Investigate fault-handling schemes at each layer, including but not limited to examining all known fault-handling schemes at each layer. This study includes:

- the quantitative performance measures of each fault-handling scheme;
- the coverage of each fault-handling scheme;
- the important assumption each scheme makes about upper and lower layer fault-handling schemes. These assumptions have implications to how well each scheme works with others, and how much it affects the combined performance measures of a fault-handling strategy.

S3. Develop an algorithm to compute the combined performance measures of a fault-handling strategy. Parameters include the performance measures of the component fault-handling schemes at each layer and how these schemes are activated.

S4. Develop an algorithm to select a fault-handling strategy that would satisfy the service requirements of a channel. The solution space could be large, but this is the stepping stone to the next step.

S5. Find an optimal or suboptimal fault-handling strategy. An optimal fault-handling strategy would meet the service requirements of the channel while incurring minimal cost. Where possible, we will develop the most efficient algorithm which will find an optimal strategy. In the case that searching for an optimal strategy is too costly (in terms of time and space), an algorithm that finds a suboptimal strategy would be developed instead.

3 Performance Measures

In a data communication channel, each layer of the protocol stack has its own way of handling errors that are detected by the protocol or reported by the lower layers.

The type of fault-handling scheme adopted by a protocol may very likely depend on the types of services that the channel is supporting, and the Quality of Service (QOS) that the network application requires. In some cases, the protocol is required to mask certain faults from the lower layers. In other cases, it will have to report the detection of the fault to the upper layer protocols immediately without taking any corrective actions.

In a real-time communication channel, the network has very tight control over the operation and use of the network resources in order to provide guaranteed end-to-end delivery delay and other QOS measures. Thus, adopting an appropriate fault-handling strategy for the channel is instrumental in the provision of this guarantee. Particularly, the latency incurred by fault-handling schemes will add to the delivery delay of each real-time packet. This means that the performance of the fault-handling schemes of each protocol has to be predictable. It may also mean that the fault-handling schemes could be coordinated so that an optimal fault-handling strategy could be designed to meet the QOS requirements of the particular channel.

We will first look at the performance measures one may use to describe a fault-handling scheme.

Fault/error types and coverage: This is the type of faults/errors that the scheme will handle and its coverage. For each type of fault or error, its coverage is defined as the probability that the fault/error will be detected by this fault-handling scheme. The specific type of errors will also depend on the particular layer of the protocol. For example, at the network layer, this may include (i) bit errors in data/payload portion of the packet, and (ii) a misrouted packet (address error in the header, leading to a process called *cell insertion* in ATM networks). The related parameters are types of faults, rate of fault occurrence, percentage of faults covered, and fault detection probability.

Latency: This is the time overhead incurred by a fault-handling scheme. It could vary widely or even be unboundable, depending on the fault-handling scheme. The related parameters are the underlying detection mechanism, diagnosis process, retry/reconfiguration time, and restart time (if any).

Utilization: The utilization is expressed as a percentage of the available bandwidth that has been actually used to process the data from the upper layer. The related following parameters include the ECC/checksum scheme, recovery algorithm, and diagnosis mechanism.

Buffer: The maximum buffer size required by the scheme. This may be expressed in terms of number of data frames.

Cost: This could include extra hardware required, and any other costs not accounted for.

In a real-time communication channel, these performance measures could also indicate the target performance measures that the fault-handling scheme must meet. This is particularly true for the latency incurred by the scheme. A bound on the latency means a restriction on the amount of processing allowed, and the number of retries that the scheme may use.

4 SAFENET

Future generations of computer systems are expected to handle more and more time-critical, mission-oriented, and distributed applications. Particularly in a military setting, the ability of the network to support fault-tolerant real-time communications becomes an acute issue, as well as a challenging engineering problem. Thus, the recently approved Survivable Adaptable Fiber Optic Embedded Network (SAFENET) Standard [8, 9], the underlying network architecture of the Navy Next Generation Computer Resources (NGCR) Program [10], has been chosen for our in-depth study of the proposed layered approach.

4.1 Overview of SAFENET

SAFENET is a computer network standard developed by the Department of Defense based on the open system concept [8, 11]. It is capable of using both military and readily available commercial products that are interoperable with each other.

SAFENET is a layered architecture that corresponds closely to the OSI Reference Model. In the SAFENET model, the entire architecture is partitioned into three broad layers: SAFENET User Services, SAFENET Transport Services, and SAFENET LAN Services.

At the top is the SAFENET user who interacts directly with the SAFENET User Services. The services provided here would include file transfer and network management applications. These services correspond to those provided in the Application, Presentation and Session Layers of the OSI Model.

In the middle is the SAFENET Transfer Services which are primarily concerned with the delivery of data from one end to the other. Its responsibilities include connection management, packet routing, and segmentation and reassembly of packets. These services are similar to those provided by the Transport, Network and Logical Link Control (upper sublayer of the Datalink layer) in the OSI Model.

At the bottom is the SAFENET LAN Services. This layer corresponds to the Medium Access Control layer (lower sublayer of the Data Link layer) and the Physical layer of the OSI Model.

SAFENET is also broken down vertically by protocol suites. The architecture employs the OSI protocol suite and the Lightweight protocol suite [8] as well as a combination of both. The OSI protocol suite will be used when interoperability is paramount, and the Lightweight protocol suite will be used when performance is the driving factor.

We will examine the fault-handling schemes of SAFENET associted with LAN and Transfer Services, and evaluate the performance of these schemes. We will also discuss briefly how faults are handled in the User Services Layer. Fig. 2 is a layered view of the fault-handling schemes in SAFENET.

4.2 LAN Services

SAFENET's LAN Services are provided entirely by FDDI, and thus rely on the FDDI standard to provide the fault-tolerance measures at the MAC layer and below. FDDI has a sophisticated suite of fault-detection and error-recovery measures that can handle many of the errors that may surface at this layer [12–14]. Besides its many error-detection capabilities through the various parameters in the data frames, it can also detect a topology error (e.g., node and link failures) and reconfigure itself to isolate the faulty component.

4.2.1 Claim and Beacon Processes: The Valid Transmission Timer (TVX) of each station acts as a watchdog timer for data transmission. It is reset to a specific time period each time the station receives a valid token or a valid frame. When the TVX timer expires, or when the error counter reaches a certain threshold, the station enters the Claim Process by continuously transmitting a "claim frame". Whenever a station that is not already in the Claim Process receives a claim frame from another station, it too enters the process by continuously transmitting a claim frame. Each claim frame carries a "bid" from the originating station for the value of the Target Token Rotation Time (TTRT). When a station receives a claim frame, it compares the bid with its own bid and then only repeats the frame if it has a lower bid. If a station receives its own claim frame, then it has the lowest bid for the TTRT, and hence, issues the initialization token so that each station on the ring can set its own TTRT.

At the start of the Claim Process, the Token Rotation Timer (TRT) of each station is set to a maximum value, T_Max, which is settable by the user. If the TRT expires during the Claim Process, then there may be a change in the topology and the ring will enter the Beacon Process. The Beacon Process uses a "beacon frame" to traverse the ring in an attempt to detect the break in the ring. If a station receives its own beacon, then the ring is all right and the stations enter the Claim Process again.

Claim Process: Consider a FDDI station A that first initiates the Claim Process in the ring. If the Claim Process is entered upon expiration of the TVX timer, then the first manifestation of the fault as an invalid transmission must have surfaced between the last valid transmission and the invocation of the Claim Process.

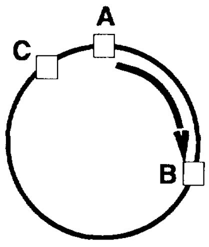


Figure 3: A simple logical view of a FDDI ring

Suppose station A in Fig. 3 transmits a Claim Frame with its bid for the TTRT. If it has the lowest TTRT, then it will find its own Claim Frame returned to itself after one rotation through the ring. This should happen within the time stored in the TRT.

Now, all other stations on the ring enter the Claim Process *after* Station A. If Station A does not have the lowest bid, then the bidding process will end only after some other station on the ring received its own Claim Frame. Assume that Station C is the immediate upstream neighbor of Station A. In the worst case, Station C enters the Claim Process only after receiving the Claim Frame from Station A and it issues the lowest bid for the TTRT. Therefore, Station C wins the bidding process after a maximum of 2 TRT $-\epsilon_{AC}$ seconds, where ϵ_{AC} is the transmission time between Stations A and C. In a large ring with many stations, ϵ_{AC} is small relative to the TRT, so the above expression is bounded by 2 TRT.

At the end of the bidding process, the station that won the bidding passes an initialization token once around the ring. The time taken should be bounded by TRT.

Summing the bounds described above, the fault-tolerance latency for the successful Claim Process is simply TVX + 3 TRT.

Beacon Process: The Beacon Process may be initiated by the SMT or be entered upon expiration of the TRT

LAYER	FAULTS/ ERRORS	DETECTION METHOD	RECOVERY METHOD	RECOVERY TIME
PHY	break in cable	electrical	report upwards	boundable
	symbol error	4B/5B coding	report upwards	boundable
MAC	corrupted data	CRC, FCS fields	-	boundable
	Frame not copied	FS fields	-	boundable
	Addr not recognized	FS fields	report upwards	boundable
	Late/Lost token	timer TRT	Claim Process	boundable
	Lost token/frame	timer TVX	Claim & Beacon	boundable
	change in topology	timer TVX	Trace Process	boundable
LLC	Lost frames	Sequence no.	Report	-
ISO IS/IS	Lost frames	Timeout	-	-
IP	Lost frames	Timeout	-	-
TP4	No acknowledge pkt	Sliding Window	Retransmit	boundable
	Lost connection	Inactivity timer	Reconnect	boundable
CLTP	Lost frames	Acknowledgement	Report	-
TCP	Lost frames	Sliding Window	Sel. Repeat	boundable
	Lost connection	Inactivity Timer	Reconnect	boundable
SES	Unacknowledged	Timer	Resynch	-
PRE	-	-	-	-
APP	Lost connection/ lost data/ corrupted Data	Timer/Reports by lower layers/CRC/ Reasonableness check	Application dependent	Application dependent

Figure 2: Layered view of fault-handling schemes in SAFENET

during the Claim Process (failure of the Calim Process). The SMT may initiate the Beacon Process when it knows that there is a change in the topology of the ring (e.g., insertion of a new station). Since there is no general fault model that can adequately describe such a procedure, we shall only consider the case of a failed Claim Process.

The Claim Process is entered upon expiration of the TVX timer, and recall that the TRT timer is set to T_Max. The Claim Process is considered failed when a station that is transmitting Claim Frames has yet to receive any Claim Frames from its upstream neighbors upon expiration of the TRT. Again, assume that Station A first initiates the Beacon Process upon expiration of the TRT timer. Each station transmits a continuous stream of Beacon frames and yields to the frames from upstream. This continues until a station receives its own Beacon frame, or when the Stuck Beacon Timer, T_Stuck (maintained by the SMT), expires. (T_Stuck is usually set to 10 seconds.) Upon successful completion of the Beacon Process, the station will enter the Claim Process.

Figuring all of the above elements in, we can bound the fault-tolerance latency of the Beacon Process by TVX + T_Max + T_Beacon + 3 TRT, where T_Beacon represents the time required to set up the Beacon Process upon failure of the Claim Process.

4.2.2 Trace Process: The Trace Process is initiated when the Beacon Process has failed, i.e., the Stuck Beacon Timer has expired. This indicates the presence of a fault between the beaconing station and its upstream neighbor. The beaconing station uses physical signaling to send the trace message to its upstream neighbor. Both stations leave the ring and perform a ring confidence test. If the test is successful, then the Claim Process is restarted. Otherwise, the faulty station is removed through ring wrap, and the Claim Process is started again.

The reconfiguration time here is difficult to estimate without knowledge of implementation details, which are not specified in the FDDI standards. Assuming that the time taken from the initiation of the Trace Process to the completion of the Path Test is t_pathtest, we can express the upper bound of the fault-tolerance latency of the Trace Process as TVX + T_Max + T_stuck + t_pathtest + 3 TRT when the path test is successful, or TVX + T_Max + T_stuck + t_pathtest + t_remove-station + 3 TRT when the path test is unsuccessful.

4.3 Transfer Services

The SAFENET Transfer Services consist of the Logical Link Control (LLC) portion of the Data Link layer of the OSI model, the Network and Transport layers as well as the Transfer Services Interface (needed only if the services are implemented separately from the User

Services).

4.3.1 OSI Profile: In the OSI profile, the SAFENET Transfer services are provided by the ISO Transport Protocol Class 4 (TP4), the error detection and recovery class, which assumes that the underlying network services are unreliable and error-prone. (This is perhaps an over-conservative assumption for FDDI.)

OSI Profile Logical Link Control: The LLC layer provides error and flow control for three types of services: unacknowledged connectionless, acknowledged connectionless, and connection-oriented. There is no fault detection and recovery for the unacknowledged connectionless service. However, the LLC is able to detect lost packets for the acknowledged connectionless service as well as the connection-oriented service. It does so with the acknowledgement frames for the former and a combination of packet sequence numbers and acknowledgement frames for the latter.

The LLC reports all cases of anomaly to the upper layers and assist the upper layers in fault recovery by sequencing data packets. Thus, the fault-handling latency in this layer is deterministic and represented as $t^{\ell}_{LLC-Ack-CL}$ for the acknowledged connectionless service, or t^{ℓ}_{LLC-CO} for the connection-oriented service.

OSI Profile Network Layer: The Network layer similarly detects lost packets and misrouted packets, and reports to the Transport layer. In addition, it also checks for corrupted packets using the checksum and lifetime count in each packet header. Depending on the type of channel established, it also checks for the Quality of Service (QOS).

Like the LLC, the Network layer is provided with minimal fault-handling capabilities. Hence, the time incurred to implement the various checks is determined and denoted as t^{ℓ}_{Ntwk}. This feature is implemented in the OSI profile of the SAFENET protocol stack as well as the Lightweight protocol stack. The extra bandwidth needed is also determined and denoted as B_{Ntwk} bits per packet.

OSI Profile Transport Layer:SAFENET specifies the use of ISO Transport Protocol Class 4 (TP4), the error detection and recovery class, in the Connection-Oriented Transport Layer (COTP). TP4 uses data acknowledgement, checksums, and four timers to detect various types of errors/faults. Of particular interest is the Automatic Repeat Request (ARQ) protocol that is used for error recovery and flow control. But, we will first give an overview of the throughput of TP4.

If a data acknowledgement is sent once every N data packets, then the extra bandwidth needed to send the acknowledgement is $\frac{B_{pkt}}{N}$ bits per packet, where B_{pkt} is the length of each acknowledgement packet. The error latency would then be lower-bounded by $(N+1)\cdot\frac{B_{pkt}}{W}$ where W is the bandwidth allocated[1] to the associated channel. The bandwidth-cost of this error recovery is high: up to $N \cdot B_{pkt}$ bits have to be retransmitted per error detected.

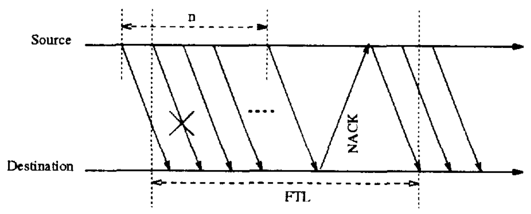


Figure 4: Fault-Tolerance Latency in ARQ Protocol.

If NR_{max} is the maximum number of retransmissions at the Transport layer, then the additional bandwidth required would not exceed $NR_{max}\cdot(N+1)B_{pkt}$. Let P_{LF} indicate the probability of a frame loss, then the additional bandwidth needed would be $P_{LF}\cdot NR_{max}\cdot(N+1)B_{pkt}$ on the average.

TP4 uses the ARQ protocol (or sometimes known as the Sliding Window protocol [6]) to recover from lost frames and perform flow control. The receiver sends back an explicit acknowledgement frame with a request number for the next frame in sequence and a "window" size. The window size determines how many frames can be accepted by the receiver beyond the frame requested.

In the ARQ Go-Back-N (ARQ-GBN) scheme, the sender may send up to n frames before repeating frames that have not been received correctly, where n is the window size. In addition, let C_{pkt} be the time needed to transmit a frame, P_{pkt} be the propagation delay from the source to the destination, N be the maximum number of retries, and T be the timeout value. Since we are considering TP4 over FDDI, we can consider the TTRT as the round-trip time from the source to the destination. The fault-tolerance latency of the ARQ scheme is thus

$$\begin{aligned} FTL &= nC_{pkt} + (TTRT - C_{pkt}) + C_{pkt} \\ &= nC_{pkt} + TTRT \end{aligned}$$

4.3.2 TCP/IP Profile: The SAFENET TCP/IP profile includes the IEEE 802.2 LLC, the DARPA Internet Protocol (IP) [15], and the DARPA Transfer Control Protocol (TCP) [16] at the Transfer Services layer, and Simple Network Management Protocol (SNMP), Domain Name Service (DNS) and the Network Time Protocol (NTP). The Transfer Control Protocol and Internet Protocol (popularly referred to as TCP/IP) are the underlying protocol suites of the successful and rapidly expanding Internet. The two protocols have some built-in

[1]at the time the channel was established

mechanisms to facilitate error detection and flow control. They are in many ways very similar to the ISO protocols. In fact, the TCP/IP protocol suite was developed before the ISO protocols, and the latter has assimilated many of the features of the former.

TCP/IP Profile Logical Link Control: This part of the profile is defined by the IEEE 802.2 standard, although SAFENET only uses LLC Type 1, the connectionless protocol. LLC Type 1 has minimal fault recovery, which is left to the upper-layer protocols to accomplish.

TCP/IP Profile Network Layer: At the Network Layer, the services are provided by the IP which is very similar to the ISO CLNP; that is, the IP is also a connectionless protocol, and leaves fault recovery to the upper-layer protocols. So, it incurs minimal processing overhead in handling errors/faults.

TCP/IP Profile Transport Layer:TCP is similar to the ISO TP4; in fact, TP4 uses many approaches of TCP. Like TP4, TCP has various mechanisms to detect and handle errors that could occur in the data transmitted and the connection between two hosts. TCP uses a version of ARQ that is very similar to the Selective Repeat ARQ protocol. As in the general ARQ, each data frame is identified with a Sequence Number, and the recipient returns a Request Number that is piggybacked onto a data frame in the reverse direction. If there is no data in the reverse direction, then only the Request Number is returned in the form of an acknowledgement.

Let n be the window size of the ARQ scheme as before, C_{pkt} be the time needed to transmit a frame, P_{pkt} be the propagation delay from the source to the destination, N be the maximum number of retries, and T be the value of the timeout timer. Again, we are considering TCP over FDDI, and hence, the round-trip time is bounded by 2 TTRT and the average round-trip time is TTRT. Thus, the latency of retransmission in TCP is bounded by

$$\begin{aligned} FTL &= nC_{pkt} + (TTRT - P_{pkt}) + P_{pkt} \\ &= nC_{pkt} + TTRT \end{aligned}$$

which is the same as in TP4.

4.3.3 Lightweight Profile: The Lightweight Profile is designed for applications that require time-sensitive or time-critical data transfers. The Lightweight Profile is primarily made up of the OSI LLC Type 1 (CL) and the Xpress Transfer Protocol (XTP) [17].

Lightweight Profile Logical Link Control: As is the case in the other profiles, the services in this layer is provided by the ISO LLC Type 1 Connectionless service, which does not have any fault-recovery mechanisms.

Lightweight Profile Network and Transfer Layers: Data transfer services at the Network and Transport layers are handled by XTP, a protocol developed for serving time-sensitive and time-critical applications on a best-effort basis. Some of its features include [17]:

1. implicit connection setup initiated by the client,
2. switch-like routing with a mechanism to set up and tear down connections,
3. message-priority/scheduling with a priority field in each packet header,
4. reliable multicast mechanisms (*bucket*, *damping* and *slotting* heuristics to reduce bandwidth requirement,
5. real-time reliable datagram capability using an immediate acknowledgement policy,
6. both flow and rate control,
7. selective retransmission and acknowledgement.

XTP also has fault-handling features similar to TCP and TP4. It can perform Selective-Repeat and Go-Back-N ARQ, though the user has the option to turn this feature off to avoid the high overhead associated with the processing necessary to provide reliable services. In addition, like TP4, the receiver may exercise end-to-end flow control by specifying the maximum window size that the sender may use. Therefore, the fault-tolerance latency of XTP is the same as that of TP4 and TCP.

4.4 User Services

The SAFENET User Services include the top three layers of the OSI model: Session, Presentation and Application, plus the application interface and the SAFENET Time Services (STS).

The Session layer fault detection and handling depends on whether a Service Protocol Data Unit (SPDU) is confirmed or unconfirmed. Confirmed services expect an acknowledgement from the other end of the connection, while an unconfirmed service does not. Upon detection of faulty data, the Session layer may abort the session or proceed with resynchronization. Again, this depends on whether the service is confirmed or unconfirmed.

The Presentation layer has little fault detection and reporting capabilities. However, it may have provisions for the network service provider or the user to initiate an abortion of the connection.

The fault detection and handling capabilities at the Application layer depend entirely on the underlying application, the type of service, and the QOS that the user requires. Sophisticated fault handling is possible.

5 Fault-Handling Strategy

A fault-handling strategy is defined as a set of fault-handling schemes working together over a protocol stack. Once we decide on the performance measures of a fault-handling scheme at each layer of the protocol stack, we can compute the combined performance measure of the schemes used in the strategy. The combined performance measure of a strategy will then help us predict its capability in meeting the performance and dependability requirements as well as the amount of resources it will require for its implementation.

5.1 Combined Performance Measure

The combined performance measure of a fault-handling strategy is computed from the performance measures of each fault-handling scheme used in the strategy. Recall that a data frame from the upper layer could be fragmented into several packets in a lower layer, and the delay that a message experiences will depend upon the delay each of its fragments will experience.

Let f_ℓ = fault-tolerance latency at layer ℓ, T_ℓ = time for transmitting a data frame at layer ℓ, $E_{max,\ell}$ = max # of retransmissions at layer ℓ per data frame from layer $\ell+1$, and n_ℓ = max # of fragments per data frame from layer $\ell+1$. Then we get

$$\begin{aligned} f_{\ell+1} &= E_{max,\ell} f_\ell + n_\ell T_\ell \\ f_m &= \textstyle\sum_{i=1}^{m-2} n_i T_i (E_{max,i})^{m-i} + (\prod_{i=1}^{m-1} E_{max,i}) f_1. \end{aligned}$$

We may now proceed to compute the combined performance measures of a fault-handling strategy. Let Q_ℓ denote the set of fault-handling schemes at the layer $\ell \in \{1, 2, \ldots, m\}$, $P_\ell(q)$ represent the performance measures of the fault-handling scheme $q \in Q_\ell$, and $v_\ell^\epsilon(q)$ represent the fault coverage of the fault-handling scheme q, in the layer ℓ, for the fault/error ϵ. Then $P_\ell(q)$ is a 5-tuple $< v, f, w, B, C >$ where v is the fault coverage, f is the fault-tolerance latency, w is the utilization, B is the maximum buffer size required, and C is the processing cost.

Let us define the fault-handling strategy $\mathcal{S}$ as a set over $\bigcup_\ell Q_\ell$. Then the combined performance of $\mathcal{S}$ is $P(\mathcal{S}) \equiv < v_\mathcal{S}^\epsilon, f_\mathcal{S}, w_\mathcal{S}, B_\mathcal{S}, C_\mathcal{S} >$ where $v_\mathcal{S}^\epsilon = \max_{q \in \mathcal{S}} v_q^\epsilon$, $w_\mathcal{S} = \min_{q \in \mathcal{S}} w_q$, $B_\mathcal{S} = \sum_{q \in \mathcal{S}} B_q$, $C_\mathcal{S} = \sum_{q \in \mathcal{S}} C_q$, and

$$f_\mathcal{S} = \sum_{i=1}^{m-2} n_i T_i (E_{max,i})^{m-i} + (\prod_{i=1}^{m-1} E_{max,i}) f_1.$$

5.2 Determination of Schedulability

Given a fault-handling strategy for a real-time channel, we have to determine if it meets the client service requirement, and thus the schedulability of the real-time channel.

The client service requirement $\mathcal{R}$ of a real-time channel is a set of the desired performance measures of the real-time channel. The set elements are 5-tuples $< v, f, w, B, C >$ as defined above. Each set element represents an acceptable performance measure of the channel.

To compute the combined performance measures of a real-time channel, we propose the following two steps:

S1. Given the client service requirement $\mathcal{R}$ of a real-time channel and the fault-handling strategy $\mathcal{S}$, compute the combined performance measure of $\mathcal{S}$, $P(\mathcal{S})$.

S2. Determine if the real-time channel is schedulable. It is schedulable if $\exists r \in \mathcal{R}$ such that $v_\mathcal{S}^\epsilon \geq v_r^\epsilon$, $w_\mathcal{S} \geq w_r$, $B_\mathcal{S} \leq B_r$, $C_\mathcal{S} \leq C_r$, and $f_\mathcal{S} \leq f_r$.

5.3 Construction of a Strategy

In order to establish a real-time channel over a protocol stack, one must select a fault-handling strategy that meets the client service requirements. The selection of the strategy could be performed by the client prior to requesting the establishment of a channel, or by the network manager in response to a request.

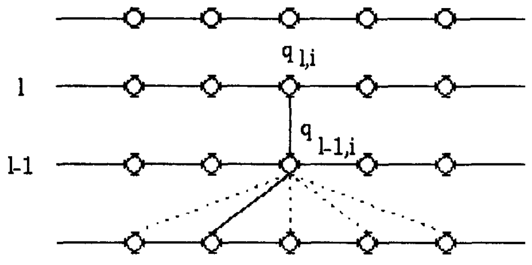


Figure 5: Selecting an optimal fault-handling strategy.

The algorithm we describe below uses a dynamic programming approach to select an optimal strategy. Recall that a schedulable strategy is one whose Combined Performance Measures (CPM) meets the client service requirements.

Since the performance measure is a 5-tuple, we can define only a partial ordering on the set of performance measures. Particularly, we say that a strategy is optimized with respect to (w.r.t.) a certain measure if it has the best performance in that measure.

Note that at any layer ℓ, given a selected set of fault-handling schemes at *that* layer and those *below*, we can compute the CPMs of that set, thus enabling comparisons of two different sets of fault-handling schemes at layer ℓ. Therefore, for two different set of fault-handling strategies $\mathcal{S}_1$ and $\mathcal{S}_2$, with CPM($\mathcal{S}_1$) and CPM($\mathcal{S}_2$) respectively, we say that CPM($\mathcal{S}_1$) $\preceq$ CPM($\mathcal{S}_2$) w.r.t. f if $\mathcal{S}_1$ passes the schedulability test and $f_{\mathcal{S}_1} < f_{\mathcal{S}_2}$.

In the algorithm given below, there are L layers in the protocol stack, 1 being the lowest and L being the highest. Q_ℓ is the enumerated power set of the set of fault-handling schemes at layer ℓ. $P_\ell[i]$ is an array that keeps track of the best possible selection so far involving fault-handling scheme i at layer ℓ.

Algorithm A: Selection of an optimal strategy

```
initialize array $P_\ell[i]$ to $\infty$ for $\ell = 2...L$ and
$i = 1...||Q_\ell||$ initialize array $P_1[i]$ to CPM($q_{\ell,i}$)
for $i = 1...||Q_1||$
for $\ell = 2$ to $L$
    for $i = 1$ to $||Q_\ell||$
        for $j = 1$ to $||Q_{\ell-1}||$
            $\rho \leftarrow$ CPM($q_{\ell,i}, q_{\ell-1,j}$)
            if ($\rho \preceq P_\ell[i]$) then $P_\ell[i] := \rho$
        end_for
    end_for
end_for
select $\min_i P_L[i]$
```

5.4 Simulation Experiments

To determine the tradeoffs between the various performance measures, we ran simulations based on Algorithm A. We selected a set of client requirements and a set of performance measure figures from the SAFENET protocol stack. The simulation program selects the optimal strategies with respect to the maximum fault-tolerance latencies, the buffer size and the error coverage, respectively. For each performance measure that is being optimized, the average value of the other performance measures are computed.

For example, given the various client requirements, including the maximum fault-tolerance latency, the optimal strategy is selected for each set of data describing the performance measures of the protocol stack. This strategy is optimized with respect to the fault-tolerance latency (i.e., of all the strategies that meet the client requirement), and hence, the optimal strategy has the lowest fault-tolerance latency. For the same client requirement, we select an optimal strategy for each data set. Then the average values of the error coverage, the utilization and the buffer size of all these optimal strategies are computed. The average values are then plotted against the maximum fault-tolerance latencies specified in the various client requirements. Part of the results of the example above are plotted in Fig. 6.

From Fig. 6, the average error coverage is approximately proportional to the increase in the client-required maximum fault-tolerance latency. Since the effective error coverage is computed by a *system reliability block method* (an error is not detected if all the fault-handling schemes couldn't detect it) the coverage increases as more handling schemes are involved. However, this also means an increase in the fault-tolerance latency, which accounts for the relation described by Fig. 6.

Using a similar argument, the average buffer size required increases linearly with the maximum fault-tolerance latency allowed, as more fault-handling schemes are included in a fault-handling strategy, their total buffer size increases. Much of the buffer requirement comes from the sliding window schemes for flow and error control; an increase in the window size means an increase of an integral multiple of the maximum packet size.

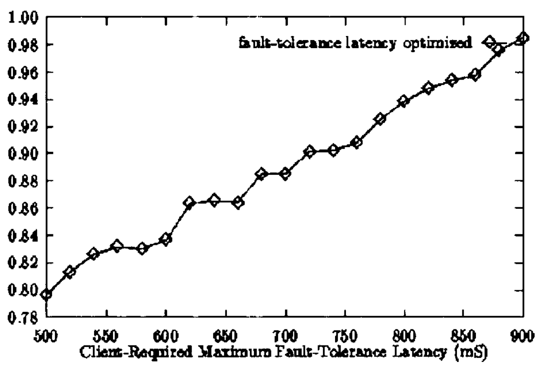
On the other hand, the channel utilization can be expected to drop as more fault-handling schemes are involved. The decrease in channel utilization in a fault recovery is attributable to the additional bandwidth needed for retransmissions and acknowledgements. This is especially costly in a sliding window scheme using Go-Back-N with a large window size. This also accounts for a larger fault-tolerance latency.

6 Conclusion and Future Work

Our approach to fault-tolerant real-time communications is based on a layered view of the fault-handling schemes in the communications architecture. This approach is consistent with many existing protocol stacks. With this approach, we can evaluate the performance of each fault-handling scheme at each layer in terms of certain quantitative performance measures, such as fault coverage and latency. Using these measures, we have also shown how to construct a fault-handling strategy by selecting a suitable set of fault-handling schemes at the various layers so that they can collectively meet the client's performance/dependability requirements.

We have determined the performance measures of a fault-handling scheme, including fault coverage, fault-tolerance latency, bandwidth utilization, the buffer size required and the other costs. Using the performance measures of each scheme, we developed an algorithm to compute the combined performance of a fault-handling strategy, and thus, an algorithm to select a fault-handling strategy that will meet the service requirements of a channel.

SAFENET is used as the main vehicle of our study, though the proposed approach can apply to any layered protocol. From a thorough review of the fault-handling features of SAFENET, we found that SAFENET has many fault-handling schemes at each layer (LAN Services, Transport Services, User Services) corresponding to each protocol suite (OSI, Lightweight, Mixed and TCP/IP). Its versatility and its adoption of various existing protocols and standards make it an interesting architecture for the analysis with respect to fault-tolerance and timeliness.



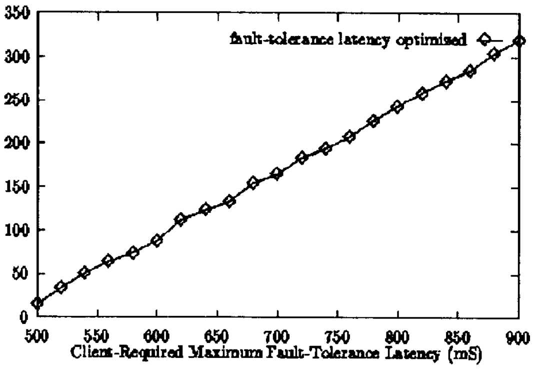


Figure 6: Average error coverage and buffer size vs. client-required fault-tolerance latency.

Aside from SAFENET, we have been investigating other fault-tolerance and performance measures, and evaluate other fault-handling schemes and protocols. This includes various fault-handling schemes in other multi-access networks [18] as well as point-to-point networks [19]. Since these two classes of networks are quite different in nature at the lower layers, it should make very interesting comparisons.

References

[1] D. D. Kandlur, K. G. Shin, and D. Ferrari, "Real-time communication in multi-hop networks," *Proc. 11-th Int'l. Conf. on Distributed Computing Systems*, pp. 300–307, 1991.

[2] D. Ferrari and D. C. Verma, "A scheme for real-time channel establishment in wide-area networks," *IEEE Journal on Selected Areas in Communications*, vol. SAC-8, pp. 368–379, Apr. 1990.

[3] D. Ferrari, J. Ramaekers, and G. Ventre, "Client-network interactions in quality of service communication environments," in *Proceedings of 4th IFIP Conference on High Speed Network*, IFIP, December 1992.

[4] J. Kurose, "Open issues and challenges in providing quality of service guarantees in high-speed network," *Computer Communication Review*, vol. 23, pp. 6–15, January 1993.

[5] Q. Zheng and K. G. Shin, "On the ability of establishing real-time channel in point-to-point packet-switched networks," *IEEE Trans. on Communications*, vol. 42, no. 2/3/4, pp. 1096–1105, Feb./Mar./Apr. 1994.

[6] D. Bertsekas and R. Gallager, *Data Networks*. Prentice Hall, 2nd ed., 1992.

[7] A. Tanenbaum, *Computer Networks*. Prentice Hall, 2nd ed., 1989.

[8] Department of Defense, "SAFENET – Survivable Adaptable Fiber Optic Embedded Network," *Military Standard MIL-STD-2204 (Advanced Copy)*, September 1992.

[9] Department of Defense, "SAFENET network development guidance," *Military Handbook MIL-HDBK-818-1 (Advanced Copy)*, September 1992.

[10] Department of Defense, "NGCR system fault tolerance report (draft)," *NGCR Fault Tolerance Task Group*, August 1993.

[11] J. L. Paige and E. A. Howard, "SAFENET II – the Navy's FDDI-based computer network standard," in *FDDI, Campus-Wide, and Metropolitan Area Networks* (K. Annamalai, S. K. Cudworth, and A. B. Kasiewicz, eds.), pp. 7–13, SPIE – the International Society for Optical Engineering, September 1990.

[12] X3T9.5/88-139, *FDDI Media Access Control (MAC-2) working draft proposed American National Standard.* Computer & Business Equipment Manufacturers Association, September 3, 1992.

[13] X3T9.5/84-49, *FDDI Station Management (SMT) draft proposed American National Standard.* Computer & Business Equipment Manufacturers Association, June 25, 1992.

[14] S. Mirchandani and R. Khanna, eds., *FDDI Technology and Applications.* John Wiley & Sons Inc., 1993.

[15] Information Sciences Institute, "RFC 791 – Internet Protocol," *Internet Engineering Task Force*, 1981.

[16] Information Sciences Institute, "RFC 793 – Transmission Control Protocol," *Internet Engineering Task Force*, 1981.

[17] Protocol Engines, "XTP protocol definition revision 3.6," Tech. Rep. PEI 92-10, Protocol Engines Inc, 11 January 1992.

[18] C. C. Chou and K. G. Shin, "Statistical real-time channels on multiaccess networks," *submitted for publication*, July 1994.

[19] Q. Zheng, *Real-Time Fault-Tolerant Communication in Computer Networks.* PhD thesis, The University of Michigan, 1993.