A FAULT-TOLERANT BRIDGING SCHEME
FOR DYNAMIC LOAD BALANCING

Tsang-Ling Sheu
IBM Corp.
Networking Systems
Research Triangle Park, NC 27709

ABSTRACT

This paper presents a fault-tolerant bridging scheme for interconnecting multiple LANs (Local Area Networks). Fault tolerance is built by linking a LAN to two bridge ports located on two different processing elements, respectively. The proposed bridge can provide better performance by balancing the traffic between a high-speed LAN and a low-speed LAN. Two load-partitioning schemes are developed to prevent multiple copies of an arriving frame. A static partitioning scheme requires little hardware or software augmentation. However, the load may not be balanced due to non-uniform traffic patterns. On the other hand, a dynamic partitioning scheme can offer dynamic load balancing, although it requires mechanisms to monitor the traffic and to synchronize interprocessor communication. The bridge performance in terms of frame response time is evaluated through simulation. In the simulation, impacts of the two partitioning schemes on bridge performance are compared. Finally, interprocessor synchronization cost is analyzed. It is concluded that the dynamic partitioning scheme can contribute better performance as long as the synchronization cost is lower than 45% of the processing requirement for a frame.

Index terms: Bridging, Load Balancing, Fault Tolerance, LANs.

1. INTRODUCTION

This paper presents a fault-tolerant bridging scheme to offer dynamic load balancing for extended local area networks (LANs). A bridge, defined here to operate at the logical link or media access control layer [1], is a device that links two or more LANs and allows stations to connect to different LANs. Two different bridging approaches, source routing and transparent bridging, were standardized by IEEE 802.5 and 802.1, respectively. A source routing scheme allows stations to communicate with multiple LANs by having the originating station specify the route [2−4]. On the other hand, a transparent bridge explicitly specifies the route for an arriving frame [5,6]. Comparisons of the two LAN bridges can be found in [7,8].

Two major issues, which directly impact the design of a bridge, are performance and fault tolerance. As a network keeps growing, a multiple-port bridge that interconnects more than two LANs will attract more and more attention. This is because a multiple-port bridge can substantially reduce the frame latency time by decreasing the number of network hops. However, compared to a two-port bridge, a multiple-port bridge may become a network bottleneck if its processing power is not increased. Contention for a bridge will seriously degrade network performance, particularly when the speed of LANs exceeds 100 Mbps, such as the FDDI (Fiber Distributed Data Interface) [9−10] and the Gigabit LANs [11]. In addition to the demand of high performance, fault tolerance is another critical issue in bridge design. The spanning-tree routing protocol [6,12], used in today's transparent bridges, guarantees data integrity through an existing alternative path. However, a weak point of this protocol is in its high cost of reconfiguration time [13]. This also includes the cost of reestablishing routing tables and resending blocking messages. The high reconfiguration cost may significantly increase frame latency time and create network congestion. Also, a second path may not actually exist to support fault-tolerant routing.

This paper therefore presents a bridge to support the communication among multiple high-speed LANs. A bridge in this design consists of a number of processing elements communicating through a switching fabric. Each processing element can effectively support two bridge ports. Through the bridge ports, paths for LAN to LAN communications are established. A fault-tolerant feature is embedded by linking each LAN to two bridge ports located on two processing elements, respectively. Once a processing element fails, bridging functions can be performed by another processing element. In addition to the concern of fault tolerance, a bridge may suffer performance degradation due to unbalanced traffic from heterogeneous LANs (Token Ring, Ethernet, FDDI, etc.). The proposed bridge can offer better perform-
ance by providing dynamic load balancing (e.g., by evenly balancing traffic from two LANs, an extremely high-speed and an extremely low-speed LAN).

As can be seen, the proposed bridge can provide advantages of better performance and fault tolerance. However, one should be aware of multiple copies of a frame being generated when a single LAN is linked to two bridge ports. Multiple copies of a frame should be avoided in interconnected LANs, since this will create severe network congestion [14–15]. This paper presents load-partitioning schemes, referred to as address partitioning (AP), to prevent multiple copies. AP can be done by the bridge itself and requires no modification of current IEEE 802 protocols. Two types of AP, a static (SAP) and a dynamic (DAP) scheme, are developed in this paper. The weak point of a SAP is that station addresses have to be partitioned and assigned to processing elements initially. This limits the capability of providing load balancing dynamically. On the other hand, a DAP assigns station addresses to each processing element based on the arriving traffic. Performance comparisons of these two load-partitioning schemes were accomplished through RESQ (RESearch Queueing Package) [16] simulation. Simulation assumed that the bridge interconnects multiple LANs with various media speeds. The simulation results indicated that a DAP can significantly reduce frame response time, particularly when high-speed LANs and low-speed LANs were interconnected through the bridge.

The rest of this paper is organized as follows. In Section 2, the architecture of a fault-tolerant bridge is presented. Section 3 introduces two load-partitioning schemes for the bridge. In Section 4, impacts of the load-partitioning schemes on bridge performance are evaluated and the simulation results are presented. Section 5 gives the concluding remarks.

2. A FAULT-TOLERANT BRIDGE

A fault-tolerant bridge is designed to guarantee data integrity for bridged LANs. One of the innovations is that traffic from unlike LANs with various media speeds can be evenly balanced. A general configuration of the bridge is presented below.

2.1. System Architecture

The proposed bridge for multiple-LAN intercommunications consists of a number of identical processing elements. Figure 1 shows a general configuration of this bridge. Each processing element, $P_E_i$, (1 ≤ $i$ ≤ $n$ where $n$ is an even integer), consists of a processor ($P$), a local memory ($M$), and two I/O ports ($B_{1P}, B_{2P}$). These two I/O ports, referred to as, respectively, a primary bridge port and a secondary bridge port, are individually linked to a LAN. As marked by "P" (primary) and "S" (secondary) in the figure, $B_{1P_i}$ ($B_{2P_i}$) is the primary bridge port of $P_E_i$, when $i$ is odd (even). $B_{2P_i}$ ($B_{1P_i}$) is the secondary bridge port of $P_E_i$, when $i$ is odd (even). All the PEs are interconnected through a switching fabric. For the purpose of fault tolerance, the switching fabric can be implemented as a multiple bus [17] or a multistage interconnection network [18]. Both of the interconnection schemes can provide multiple paths for interprocessor communication. To maintain end-to-end data integrity, each LAN is linked to a primary and a secondary bridge port located on different PEs. As shown in Figure 1, once a PE fails, frames delivered from a LAN can acquire the bridging services of the other PE.

Operations of the bridge are described through an example. As shown in the figure, a source station attached to LAN 1 delivers a sequence of frames to its destination station attached to LAN 4. Two bridge ports, $B_{1P_1}$ (primary port of $P_E_1$) and $B_{2P_1}$ (secondary
port of \( PE_2 \) will copy the frames into \( PE_1 \) and \( PE_2 \), respectively. Since multiple copies of an arriving frame are not allowed, only one processing element (say \( PE_1 \)) will process the frame and perform the frame forwarding. Methods to prevent multiple copies will be discussed in Section 3. After performing the required bridging functions, \( PE_1 \) selects paths from the multiple-path switching fabric and transmits the frame to the destination LAN (LAN 4 in this example). As can be seen, two bridge ports, \( B_2P_3 \) and \( B_2P_4 \), can be used to send out the frames. This implies that \( PE_1 \) will transmit the frame to \( PE_3 \) and \( PE_4 \) through the switching fabric. However, only one bridge port (say \( B_2P_4 \)) is selected to output the frame to LAN 4. The selection of bridge ports ensures that the destination station only receives a single copy of each frame.

2.2. Bridging Functions

A bridge interconnects multiple LANs for the purpose of forwarding frames among the networks. In general, a frame consists of a header and a data field. The header is composed of a source and a destination address field. A bridging function can be implemented either through source routing or transparent bridging. In the case of source routing, the route is determined by the source station for each frame transmitted through one or more bridges to the destination station. A routing information (RI) field contained within each frame header is required to determine the path. The implementation of source routing is simple, since the bridge only executes matching operations of the RI field.

On the other hand, a transparent bridge has to maintain a routing table to determine the path. A routing table has three entries: 1) an address entry for maintaining all the station addresses, 2) a port entry for recording the bridge port, and 3) a timer entry for deleting aged entries. When a frame is received on a bridge port, two address comparisons are performed. First, the destination address contained in the header is compared to address entries of the routing table. If the destination address is not found in the routing table, the frame is broadcast to all the bridge ports, except the one at which it arrived. If the destination address is found, the frame is forwarded to a designated bridge port. Second, the source address is compared to address entries of the routing table. If the source address is not found, it is added into the routing table. Otherwise, the entries of bridge port and timer are updated.

The proposed fault-tolerant bridge assumes that

1. A PE can recognize a frame, either a source routing (SR) or a transparent bridging (TB), by examining the header format.
2. Each PE has a local routing table maintained in its local memory.

When the source addresses are compared to address entries of the routing table, a PE not only adds or updates the entries of its own local table, but adds or updates the entries of the remaining routing tables. The former is referred to as local table updating, and the latter as remote table updating. The purpose of remote table updating is to give LANs the appearance that the bridge maintains a common routing table. In the following, operations of the \( n \) PEs are described through a bridging procedure written in PASCAL syntax.

**PROCEDURE (Bridging):**

\[
\text{FOR all } PE_i, 1 \leq i \leq n, \text{ DO} \\
\text{BEGIN} \\
\text{examine the header format of an arriving frame;} \\
\text{IF the frame \( \in \) TB function} \\
\text{THEN} \\
\text{BEGIN (*Processing a TB frame*)} \\
\text{search the destination address;} \\
\text{IF it is not found} \\
\text{THEN the frame is broadcast} \\
\text{ELSE the frame is forwarded;} \\
\text{search the source address;} \\
\text{IF it is not found} \\
\text{THEN new entries are added} \\
\text{ELSE odd entries are updated;} \\
\text{END} \\
\text{ELSE} \\
\text{BEGIN (*Processing an SR frame*)} \\
\text{perform the matching of RI fields;} \\
\text{forward the frame;} \\
\text{END;} \\
\text{END.}
\]

3. LOAD-PARTITIONING SCHEMES

Maintaining load balancing among PEs is the key to acquiring better system performance when the work demand exceeds or is near the processing capacity of a PE. This is especially true when a multiple-port bridge is used to interconnect various LANs with a range of media speeds from tens of Mbps to hundreds of Mbps. The proposed bridge inherently offers load balancing by connecting a LAN to two bridge ports. Traffic from a LAN can be divided into two equal portions based on load-partitioning schemes. For
example, if two PEs of a multiple-port bridge are used to receive traffic from two LANs, a 100-Mbps FDDI and a 10-Mbps Ethernet, the total traffic from the FDDI and the Ethernet is partitioned into two equal portions. By assigning each portion to a PE, instead of handling 100 Mbps, a PE is only required to process 55 Mbps.

Load-partitioning schemes are required for the proposed bridge, since multiple copies of a frame has to be avoided in bridged LANs. Two major issues are concerned in partitioning the traffic from a LAN. First, the IEEE 802.1 and 802.5 protocols have to be followed. This indicates that the header formats and routing tables should not be modified. Second, it is preferable that load partitioning is performed by the bridge itself instead of by the end stations. In the following, two load-partitioning schemes, SAP and DAP, are introduced. Both of them are based on a partitioning of station addresses.

3.1. Static Address Partitioning

The essential concept of static address partitioning (SAP) is to divide all station addresses into two disjoint sets, an even-address set and an odd-address set. If we let $S$ be the set of all station addresses, then we have

$$S = S_e \cup S_o,$$

where

- $S_e$ = the set of even addresses;
- $S_o$ = the set of odd addresses;
- $\phi$ = empty set.

For examples, a 4-digit station address $1234 \in S_e$ and $1235 \in S_o$. As shown in Figure 1, the $n$ PEs are in this case grouped into $n/2$ pairs (i.e., $PE_1$ and $PE_2$, $PE_3$ and $PE_4$, ..., $PE_{n-1}$ and $PE_n$). SAP initially assigns the even set ($S_e$) to one PE of a pair and the odd set ($S_o$) to the other PE of the same pair. To facilitate our presentation, in the following we assign $S_e$ to $PE_i$ and $S_o$ to $PE_{i+1}$, where $1 \leq i \leq n - 1$ and $i$ is an odd integer. Note that a frame delivered from LAN $i$ ( $1 \leq i \leq n$) has a source address ($S_e$) within the frame header.

**SAP SCHEME:**

FOR $PE_i$ and $PE_{i+1}$, $1 \leq i \leq n - 1$ and $i$ is odd, DO

BEGIN

$PE_i$ and $PE_{i+1}$ copy the frames from $B_iP_i$ ($B_iP_{i+1}$) and $B_{i+1}P_{i+1}$ ports, respectively;

IF $S_e \in S_e$

THEN

$PE_i$ performs PROCEDURE(Bridging);

$PE_{i+1}$ discards the frames;

ELSE

$PE_i$ discards the frames;

$PE_{i+1}$ performs PROCEDURE(Bridging);

END.

The address assignment of SAP is made prior to the bridge operations. Since this assignment is static, bridge design can be greatly simplified and requires little hardware or software augmentation. However, the SAP scheme indeed may not balance the LAN traffic, since it is possible that even-address stations may have more messages to deliver than odd-address stations, and vice versa. This phenomenon is referred to as non-uniform traffic patterns in this paper. To evenly balance the traffic load, a dynamic partitioning scheme is proposed as below.

3.2. Dynamic Address Partitioning

A dynamic address-partitioning (DAP) scheme can dynamically assign station addresses to a pair of PEs based on the arriving traffic patterns. It is more sophisticated than the SAP scheme in that each PE is required to monitor the traffic on its bridge ports. If the traffic load exceeds a certain threshold, a PE will start to partition its address space into a number of sub-address spaces and assign them one by one to the other PE. This address reassignment is done during the bridge operations. As a result, the traffic flow from a LAN is divided into two equal parts. In addition to traffic monitoring, load sharing between a pair of PEs requires interprocessor communication. The communication cost, to be explored in Section 4, may somewhat degrade bridge performance. Here, the subsets of $S_e$ and $S_o$ are defined as follows:

$S_e = S_{(0)} \cup S_{(2)} \cup S_{(4)} \cup S_{(6)} \cup S_{(8)}$

$S_o = \bigcup_{j \in \text{even integer}} S_{(j)}$

$S_{(k)} = \bigcup_{j \in \text{odd integer}} S_{(j)}$

where $S_{(j)}$ is the set of station addresses with the first digit being $j$ ($k$). For examples, a 4-digit station address $1234 \in S_{(0)}$ and $1235 \in S_{(2)}$.

A load threshold ($\zeta$) is defined as

$$\zeta = \frac{L_{bbp}}{V_{lan}},$$

(1)
where $L_{pbp}$ is the traffic load (in Mbps) on the primary bridge port, and $V_{lan}$ is the LAN media speed (in Mbps). Obviously, $\zeta = 50\%$ implies that traffic from a LAN has been partitioned into two equal portions. By varying $\zeta$ from 0% to 100%, Section 4 will analyze its influence on system performance. In the following, we present a scheme to maintain a 50% load threshold. Note that $SPE_x$ and $SPE_y$ are defined as two augmented address spaces handled by $PE_i$ and $PE_{i+1}$, respectively. Prior to bridge operations, we assume that

$$SPE_x = S_{e},$$

$$SPE_y = S_{o}.$$  

**DAP SCHEME:**

FOR $PE_i$ and $PE_{i+1}$, $1 \leq i \leq n - 1$ and $i$ is odd, DO

BEGIN

$PE_i$ and $PE_{i+1}$ copy the frames from $B_iP_i$ ($BPX$) and $B_iP_{i+1}$ ($BPX+1$) ports, respectively;

IF $S_x \in SPE_x$ THEN

$PE_i$ performs PROCEDURE (Bridging);

$PE_{i+1}$ discards the frames;

ELSE

$PE_i$ discards the frames;

$PE_{i+1}$ performs PROCEDURE (Bridging);

$j = 0$; (*Begin to examine the traffic load*)

$k = 1$;

IF $L_{pbp} > 50\% \times V_{lan}$, THEN

REPEAT

$PE_i$ assigns $S_{o_i}$ to $SPE_x$;

$j = j + 2$;

$PE_{i+1}$ assigns $S_{o_{i+1}}$ to $SPE_x$;

$k = k + 2$;

UNTIL

$L_{pbp} = 50\% \times V_{lan}$;

END.

As can be seen, the DAP keeps on assigning a subset of $S_x$ ($S_y$) to $SPE_x$ ($SPE_y$) as long as the load on the primary bridge port is greater than 50% of the LAN media speed. The DAP eventually divides a LAN traffic into two equal portions. One portion is assigned to the primary bridge port, and the other to the secondary bridge port. Multiple copies of a frame are prevented, since $SPE_x \cap SPE_y = \emptyset$ during any time interval.

**4. PERFORMANCE EVALUATION**

As described in Section 3, the SAP assigns station addresses to a pair of PEs prior to bridge operations. Load threshold ($\zeta$) in this case can vary from 0% to 100%, since traffic from a LAN may not be equally partitioned. On the other hand, the DAP provides the capability of dynamically assigning station addresses to a pair of PEs. As a result, $\zeta$ can converge to 50%. This section presents an evaluation which compares the influence of SAP and DAP on bridge performance.

**4.1. Evaluation Model**

We are interested in evaluating the bridge performance in terms of frame mean response time (MRT). MRT is the time that starts when a frame arrives at the input bridge port and ends when it leaves the output bridge port. MRT in fact consists of two major components: 1) the computation cost and 2) the interprocessor communication cost. The computation cost in this investigation includes:

- Examination and processing of frame headers (for both SR and TB frames);
- Performing the matching of RI fields (for SR frame);
- Searching for destination addresses (for TB frames);
- Performing local table updating (for TB frames).

The interprocessor communication cost is the cost of using the switching fabric as the communication media. It includes the cost of:

- Frame forwarding (for both SR and TB frames);
- Frame broadcasting (for TB frames);
- Remote table updating (for TB frames).

In addition to the above communication costs, the DAP scheme requires an extra cost attributed to rearranging sub-address spaces for a pair of PEs. This extra communication cost is referred to as **synchronization cost** throughout the paper. Impacts of the synchronization cost on bridge performance are analyzed via simulation.

**4.2. Simulation Results**

RESQ (RESearch Queueing Package) [16], a modeling tool developed in IBM Research Center, was used to simulate the bridge operations and its load-
partitioning schemes. RESQ can model the behavior of resource contention, such as the contention for processors, routing tables, and the switching fabric. During the RESQ simulation, the following assumptions applied:

1. A frame is always destined for a different bridge port than that it was received.
2. For an SR frame, 100 instructions are required to process its header and 50 instructions to perform the matching of RI fields. For a TB frame, 100 instructions are required to process its header, 50 instructions to search for its destination address, and 50 instructions to update the routing table.
3. The media speed of an FDDI is 100 Mbps. An Ethernet has 10-Mbps media speed and a Token Ring has 4 or 16-Mbps media speed.
4. A Token Ring uses the SR scheme, while an Ethernet employs the TB scheme. The percentages of SR and TB frames delivered from an FDDI are about the same.
5. The frame size is 64 bytes associated with a 20-byte header.
6. The switching fabric has an 800-Mbps transfer rate to maintain the utilization lower than 50%.

The fault-tolerant bridge, as shown in Figure 1, was used to interconnect various LANs with different media speeds. For example, in evaluating the performance of bridging FDDIs and Ethemets, 4 LANs were used: LANs 1 and 3 are FDDIs and LANs 2 and 4 are Ethemets. PE1 and PE2 receive the frames from LANs 1 and 2 and perform the required bridge functions. PE3 and PE4 receive the frames from the switching fabric and send them out to LANs 3 and 4.

By varying the processing power from 35 to 45 MIPS (Million Instructions Per Second), Figures 2 to 4 illustrate frame mean response time (MRT) in microseconds as a function of load threshold (ξ). ξ = 50% implies that the DAP scheme was employed such that traffic from two unlike LANs (FDDI and Ethernet in Figure 2, FDDI and 4-Mbps Token Ring in Figure 3, and FDDI and 16-Mbps Token Ring in Figure 4) can be evenly balanced. On the other hand, ξ = 100% (or ξ = 0%) imply that the total traffic from a LAN arrives at the primary bridge port (or at the secondary bridge port). In other words, ξ = 100% or ξ = 0% simply indicate that no load-partitioning scheme was used. As was mentioned in Section 3, the SAP may not really balance the traffic due to non-uniform traffic patterns. As a result, ξ can vary from 0% to 100%. Decreasing ξ from 100% to 50% or increasing ξ from 0% to 50% can considerably reduce frame MRT. In fact, ξ = 50% provides a minimum MRT, indicating that the best bridge performance is achieved. Continuing to decrease or increase ξ will increase MRT back to the same level as when ξ = 100% or 0%.
One of the observations is that the DAP provides better performance improvement when two LANs with significantly different media speeds are interconnected. As an example, Figure 3 shows the MRT variation of 100-Mbps FDDIs and 4-Mbps Token Rings interconnected by the bridge. Another observation is that the MRT curves turn out to be more acute when the processing power is decreased from 45 to 35 MIPS. This is because small processing power may be insufficient in manipulating the traffic from multiple high-speed LANs. Tables 1-9 show the utilization of \( PE_1 \) and \( PE_2 \) to further support our arguments. \( PE_1 \) (\( PE_2 \)) is the processing element that uses its primary bridge port to receive traffic from an FDDI (an Ethernet or a Token Ring). For simplicity, in the following tables we only show the utilization when \( \xi \) varies from 50% to 100%. If the utilization exceeds 90%, an "X" will be marked in the table entries.

Tables 1 to 3 show the utilization of \( PE_1 \) and \( PE_2 \) for 45, 40, and 35 MIPS of processing power, respectively. Here, the bridge interconnects two 100-Mbps FDDIs and two 10-Mbps Ethernets. As can be seen from the tables, \( PE_1 \) with 40 MIPS exceeds 90% utilization when \( \xi = 100\% \) (no load partitioning). Similarly, \( PE_1 \) with 35 MIPS exceeds 90% utilization when \( \xi \) is equal to or greater than 80%.

Tables 4 to 6 show the utilization of \( PE_1 \) and \( PE_2 \) when two 100-Mbps FDDIs and two 4-Mbps Token Rings are bridged. \( PE_1 \) with 40 MIPS will exceed 90% utilization if no load-partitioning scheme is used. \( PE_1 \) with 35 MIPS will exceed 90% utilization when \( \xi \) is equal to or greater than 90%.

Tables 7 to 9 show the utilization of \( PE_1 \) and \( PE_2 \) when two 100-Mbps FDDIs and two 16-Mbps Token Rings are bridged. \( PE_1 \) with 40 MIPS will exceed 90% utilization if no load-partitioning scheme is used. \( PE_1 \) with 35 MIPS will exceed 90% utilization when \( \xi \) is equal to or greater than 80%. As a whole, Tables 1 to 9 also reveal that the utilization of \( PE_1 \) is about the same as that of \( PE_2 \) when \( \xi = 50\% \). This is why at this point the bridge can provide the best performance in terms of MRTs.

From the above three experiments, it was demonstrated that the DAP can provide better bridge performance by balancing the traffic between a high-speed LAN and a low-speed LAN. However, the DAP requires a certain amount of interprocessor synchronization to rearrange the sub-address spaces for a pair of PEs. In the previous evaluations, we neglected to consider this extra communication cost. Nevertheless, if this expense is too high, employing the DAP scheme will become ineffectual. Figure 5 shows frame MRT as a function of the synchronization cost. In this experiment, the synchronization cost was varied from 0 to 3.5 microseconds and a 45 MIPS of processing power was used. Recall from Figures 2 to 4, MRT is about 35 microseconds for the three types of bridge connections when \( \xi = 100\% \). This experiment pointed out that if the synchronization cost is lower than 2 microseconds, the DAP scheme can cause the three types of connections to perform better than those without using any load-partitioning scheme (\( \xi = 100\% \) or 0%). As a 45-MIPS processor was used, 2 microseconds allows for 90 instructions to be executed. This cost is about 45% of the processing requirement for a TB frame (which requires 200 instructions in our simulation model).

### Table 1. Processor Utilization (45 MIPS) when Interconnecting FDDIs and Ethernets.

<table>
<thead>
<tr>
<th>( \xi ) (%)</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>( PE_1 )</td>
<td>47.77%</td>
<td>55.42%</td>
<td>63.37%</td>
<td>71.14%</td>
<td>78.91%</td>
<td>86.80%</td>
</tr>
<tr>
<td>( PE_2 )</td>
<td>47.67%</td>
<td>39.94%</td>
<td>32.03%</td>
<td>24.26%</td>
<td>16.44%</td>
<td>8.67%</td>
</tr>
</tbody>
</table>

### Table 2. Processor Utilization (40 MIPS) when Interconnecting FDDIs and Ethernets.

<table>
<thead>
<tr>
<th>( \xi ) (%)</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>( PE_1 )</td>
<td>53.70%</td>
<td>62.43%</td>
<td>71.29%</td>
<td>80.09%</td>
<td>88.95%</td>
<td>X</td>
</tr>
<tr>
<td>( PE_2 )</td>
<td>53.65%</td>
<td>44.91%</td>
<td>36.09%</td>
<td>27.36%</td>
<td>18.56%</td>
<td>9.75%</td>
</tr>
</tbody>
</table>

### Table 3. Processor Utilization (35 MIPS) when Interconnecting FDDIs and Ethernets.

<table>
<thead>
<tr>
<th>( \xi ) (%)</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>( PE_1 )</td>
<td>61.46%</td>
<td>71.48%</td>
<td>81.44%</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>( PE_2 )</td>
<td>61.29%</td>
<td>51.35%</td>
<td>40.35%</td>
<td>32.18%</td>
<td>22.30%</td>
<td>10.72%</td>
</tr>
</tbody>
</table>
Table 4. Processor Utilization (45 MIPS) when Interconnecting FDDIs and 4-Mbps Token Rings.

<table>
<thead>
<tr>
<th>ζ (%)</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>PE1</td>
<td>45.11%</td>
<td>53.36%</td>
<td>61.78%</td>
<td>70.14%</td>
<td>78.34%</td>
<td>86.85%</td>
</tr>
<tr>
<td>PE2</td>
<td>44.90%</td>
<td>36.74%</td>
<td>28.37%</td>
<td>19.99%</td>
<td>11.77%</td>
<td>3.48%</td>
</tr>
</tbody>
</table>

Table 5. Processor Utilization (40 MIPS) when Interconnecting FDDIs and 4-Mbps Token Rings.

<table>
<thead>
<tr>
<th>ζ (%)</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>PE1</td>
<td>50.64%</td>
<td>60.07%</td>
<td>69.61%</td>
<td>78.86%</td>
<td>88.24%</td>
<td>X</td>
</tr>
<tr>
<td>PE2</td>
<td>50.49%</td>
<td>41.41%</td>
<td>31.97%</td>
<td>22.55%</td>
<td>13.23%</td>
<td>5.22%</td>
</tr>
</tbody>
</table>

Table 6. Processor Utilization (35 MIPS) when Interconnecting FDDIs and 4-Mbps Token Rings.

<table>
<thead>
<tr>
<th>ζ (%)</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>PE1</td>
<td>57.86%</td>
<td>68.71%</td>
<td>79.44%</td>
<td>89.88%</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>PE2</td>
<td>58.05%</td>
<td>47.27%</td>
<td>36.52%</td>
<td>25.77%</td>
<td>15.81%</td>
<td>8.10%</td>
</tr>
</tbody>
</table>

Table 7. Processor Utilization (45 MIPS) when Interconnecting FDDIs and 16-Mbps Token Rings.

<table>
<thead>
<tr>
<th>ζ (%)</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>PE1</td>
<td>50.35%</td>
<td>57.56%</td>
<td>64.84%</td>
<td>72.15%</td>
<td>79.48%</td>
<td>86.81%</td>
</tr>
<tr>
<td>PE2</td>
<td>50.21%</td>
<td>42.99%</td>
<td>35.72%</td>
<td>28.41%</td>
<td>21.16%</td>
<td>13.89%</td>
</tr>
</tbody>
</table>

Table 8. Processor Utilization (40 MIPS) when Interconnecting FDDIs and 16-Mbps Token Rings.

<table>
<thead>
<tr>
<th>ζ (%)</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>PE1</td>
<td>56.65%</td>
<td>64.83%</td>
<td>73.16%</td>
<td>81.30%</td>
<td>89.41%</td>
<td>X</td>
</tr>
<tr>
<td>PE2</td>
<td>56.61%</td>
<td>48.48%</td>
<td>40.22%</td>
<td>31.96%</td>
<td>23.89%</td>
<td>15.23%</td>
</tr>
</tbody>
</table>

Table 9. Processor Utilization (35 MIPS) when Interconnecting FDDIs and 16-Mbps Token Rings.

<table>
<thead>
<tr>
<th>ζ (%)</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>PE1</td>
<td>64.74%</td>
<td>74.12%</td>
<td>83.47%</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>PE2</td>
<td>64.70%</td>
<td>55.31%</td>
<td>46.05%</td>
<td>36.58%</td>
<td>25.22%</td>
<td>18.12%</td>
</tr>
</tbody>
</table>

5. CONCLUSIONS

This paper has presented a fault-tolerant bridge for interconnecting multiple LANs. Fault tolerance is provided by connecting a LAN to two bridge ports located on two different PEs, respectively. A bridge designed in this fashion not only offers fault tolerance, but also provides better performance by balancing the traffic from two dissimilar LANs. Two load-partitioning schemes were proposed to prevent multiple copies of an arriving frames. The SAP scheme initially assigns station addresses to a pair of PEs. The static assignment does not require significant hardware or software modification. However, traffic from various LANs may not be evenly balanced due to non-uniform traffic patterns. Although the DAP scheme requires extra hardware to monitor the traffic.
and software for interprocessor synchronization, it does offer dynamic load balancing.

Performance analysis of the bridge was performed using RESQ simulation. In the simulation, the bridge performance in terms of frame response time was evaluated. By varying the processing power of PEs from 35 to 45 MIPS, the influence of SAP and DAP on the bridge performance was compared for three types of bridge connections. Simulation results showed that the DAP scheme can substantially reduce frame response time, especially when two LANs with significantly different media speeds are bridged. Finally, interprocessor synchronization cost created by the DAP scheme was investigated. It was concluded that the DAP scheme can contribute better bridge performance, if the synchronization cost is maintained to be lower than 45% of the processing requirement for a TB frame.

REFERENCES


