A Ring Purger for the FDDI Token Ring

Henry Yang and K. K. Ramakrishnan
Distributed Systems Architecture and Performance
Digital Equipment Corporation
550 King Street, Littleton, MA 01460.
(yang.rama@erlang.enet.dec.com)

ABSTRACT

The Fiber Distributed Data Interface (FDDI) is a 100 Mb/s token ring which includes extensive reliability and robustness mechanisms to provide fault detection, isolation, monitoring, and recovery functions. We examine two well-known problems on token rings and their potential impact on FDDI. The two problems are No-Owner Frame (NOF) and duplicate tokens. FDDI effectively resolves these problems only when either the NOF or the duplicate token collides into a transmitting station. The impact of NOFs and duplicate tokens during the period from the occurrence of the NOF and the duplicate token until they are removed by a transmitting station can be severe. This paper presents a Ring Purger that transparently and continuously removes the NOFs and detects and removes duplicate tokens.

A Ring Purger is a designated station on a ring that performs its normal functions as a station and, in addition, performs the removal of NOFs and duplicate tokens. A distributed election algorithm is used to elect a single station on a given ring to be the Ring Purger. It ensures that the ring has one and only one Ring Purger most of the time. The selection of a Ring Purger causes no disruption to the ring operation. A Ring Purger continuously performs purge operations to clean the ring of NOFs and long fragments. It also continuously detects duplicate token conditions. Once a duplicate token condition is detected, the Ring Purger initializes the ring to remove the duplicate token.

We also present performance and robustness considerations used in the design of the Ring Purger algorithms. We study the performance effect of continuously purging the ring and show that the effect is negligible even in the worst case.

1 INTRODUCTION

Local area networks (LANs) have proliferated in their use in the last decade. Their use as the intrafacility backbone serving the critical needs of corporate data communication has become commonplace. The number of users needing the services of the LAN and depending on it has also increased. The corporate communication infrastructure of LANs, Extended LANs (LANs interconnected by bridges) and wide area networks (LANs or Extended LANs interconnected by routers) may well have tens of thousands of users. Distributed applications have also grown because of the improvements in performance, flexibility and reliability of LANs. Such widespread use of LANs places a significant emphasis on the reliability, availability and robustness of the LANs and the ability to quickly manage faults in the overall network. LAN failures that cause an impact on the overall network through 'chain reactions' of failures are particularly of concern and motivate the need to have solutions which avoid such occurrences.

The advent of fiber optics and integrated circuit technology have made it possible to provide relatively low cost, high speed LAN interconnects. The Fiber Distributed Data Interface (FDDI) is a 100 Mb/s token ring. It has emerged as a new generation of commercially available LAN, following extensive standardization efforts in the American National Standards Institute (ANSI) Accredited Standards Committee X3T9.5 [4, 6, 7, 8]. It must be noted that the FDDI token ring has defined failure recovery mechanisms for a set of important error conditions - such as identifying that a token is lost by means of a Target Token Rotation Timer (TTRT) time-out and the faulty transmission of station by means of a Valid Transmission Time (TVX) time-out. For more detailed discussions on the error cases covered, refer to [6, 13]. In this paper, we focus on the need for robust algorithms to achieve correct and high performance operation on the FDDI token ring under an additional set of important error conditions not adequately considered before. We present a set of algorithms implemented by a station to enable the ring to recover from these errors.

The underlying logical topology of token rings is a closed loop structure, which inherently has the property of continuously circulating frames transmitted on the ring. Because of this, all token rings have algorithms to remove (strip) frames that have been transmitted on them. The FDDI token ring requires that an originating station be responsible to strip frames it has transmitted on the ring. When a frame is not stripped by the originator after it traverses the ring exactly once, the frame is considered to be a No-Owner Frame (NOF).

Due to a variety of error conditions, such as the failure of a transmitting station, there is a finite non-zero probability of frames not being stripped. On a busy ring, the NOF created because it was not stripped may arrive at a transmitting station currently holding the token and thus be removed from the ring. However, if the ring is relatively idle, the NOF will traverse the ring repeatedly, wasting system bandwidth at the receiving sta-
tions, and possibly cause severe congestion in stations because of the delivery of duplicates and aged frames. Delivery of such duplicates or old information is harmful since it requires considerable overhead for detecting and discarding them. An additional problem posed to stations on the ring is the circulation of long fragments on the ring. A fragment is a partial frame that has no ending delimiter. In particular, with certain network interface designs, long fragments can consume excessive system resources (bus bandwidth, memory bandwidth and compute cycles) before they are discarded.

Another error condition that causes frames to be stripped and the transient loss of frames is the existence of duplicate tokens. The duplicate token condition may result in frames being randomly lost before reaching their destination due to colliding with a transmitting station holding (one of the other token(s). In this paper, we describe a set of algorithms implemented at a station for removing NOFs and long fragments, and to detect and remove duplicate tokens.

Any of the stations on the FDDI token ring may participate in a distributed election algorithm to elect a Ring Purger. Any participating station on the FDDI token ring may be elected to be the Ring Purger. The elected Ring Purger uses a purging algorithm to remove from the ring NOFs and long fragments transparently each time a token is received by the Ring Purger. The purging algorithm takes advantage of the invariant on a correctly operating token ring which is the following: the frames transmitted by a station on the ring are the first to be received by it subsequent to capturing the token before any other frames. It can therefore strip all the frames it receives after capturing a token until it encounters any distinct 'marker' frames it has transmitted to identify the end of its transmission during a token opportunity. When the Ring Purger receives a token, it captures the token. If the Media Access Control (MAC) protocol permits, it transmits any outstanding frames. Prior to releasing the token, it starts a 'purge cycle' by transmitting two special frames called void frames, as the marker, and then releases the token. Once the purge cycle has started, the Ring Purger unconditionally removes all frames or fragments received. The purge cycle is terminated when the Ring Purger receives one of its error-free void frames, a token or a ring initialization frame. In order to increase the probability of correctly terminating the purge cycle, the Ring Purger transmits two void frames; but it terminates its purge cycle based on receiving only one error-free void. If the purge cycle is repeatedly terminated by receiving the token rather than on the reception of an error-free void, the Ring Purger then concludes that a duplicate token error condition exists and takes corrective action.

The next section describes the impact of NOFs on the performance of the ring and motivates the need for a Ring Purger on the FDDI token ring. We also describe the impact of duplicate tokens on the ring in section 2. Subsequently we describe the set of Ring Purger algorithms. These comprise the Ring Purger election, the ring purging algorithm, and the duplicate token detection and removal algorithm. In section 4, we show that having a Ring Purger has a negligible impact on the performance of the ring. Finally, in section 5 we present conclusion.

2 PROBLEM DEFINITION

2.1 NO OWNER FRAMES AND THEIR IMPACT

When a NOF traverses the ring once, then the frame is considered to have traversed the ring twice, thus potentially being delivered to the destination station(s) twice. On a busy FDDI token ring, NOFs may run into a transmitter that is currently holding the token and the NOFs are stripped without any other explicit action. However, that is true only if the ring is busy and the NOFs arrive at a station holding the token. Another situation under which existing NOFs are guaranteed to be removed is when the total amount of frames to be transmitted by one or more stations is greater than or equal to the actual ring latency. The NOFs are then guaranteed to collide into a transmitter.

It is reasonable to expect that the ring would go through busy and idle periods, when the ring is not loaded to the maximum capacity. In such a case, a NOF created during a busy period may circulate until the next busy period starts. The number of duplicates delivered to destination stations because of the NOF would depend on the length of the idle period. If this is relatively short, the effect of the NOF would be benign. However, if the idle period during which the NOF circulates is long, the effect of the NOF circulating can be severe. An idle period of say a few milliseconds in a ring with a latency of few tens of microseconds can result in the delivery of a large number of duplicates.

Consider for example a single NOF, 512 bytes long, circulating on an idle ring. In a small ring, with a total ring latency of 50 microseconds (μs), the rate at which this single NOF will be delivered to the destination station on the ring is 20,000 frames/s and the bandwidth consumed at the destination system will be 80 Mb/s. Table 1 shows the rate of reception of frames at the destination station, when the ring is idle, and there are one or more NOFs circulating on the ring. This high rate of frames arriving at the station may consume resources in the station's network interface, bandwidth on the system bus, memory subsystem, and processor cycles to determine that the frame is a duplicate and has to be discarded.

<table>
<thead>
<tr>
<th># of NOFs</th>
<th>Ring Latency (milliseconds)</th>
<th>Frame Size (bytes)</th>
<th>No. Frames received/s</th>
<th>System BW (Mbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>64</td>
<td>100000</td>
<td>51</td>
</tr>
<tr>
<td>1</td>
<td>50</td>
<td>512</td>
<td>20000</td>
<td>80</td>
</tr>
<tr>
<td>1</td>
<td>500</td>
<td>4500</td>
<td>2000</td>
<td>72</td>
</tr>
<tr>
<td>5</td>
<td>500</td>
<td>512</td>
<td>10000</td>
<td>40</td>
</tr>
<tr>
<td>10</td>
<td>500</td>
<td>512</td>
<td>20000</td>
<td>80</td>
</tr>
</tbody>
</table>

Table 1 NOF Performance Impact Example
Thus, NOFs circulating on the ring can cause an extremely large number of duplicates to be delivered to the destination stations in a very small amount of time. This can cause receiving stations on the FDDI token ring to be severely congested. In addition, when there are multiple LANs interconnected by bridges or routers, the delivery of frames at a high rate might cause these stations to be overloaded, and also cause severe congestion on the Extended LAN and eventually the wide-area network (WAN). The situation is particularly severe when there are bridges or routers to slower data links such as to an Ethernet/802.3 CSMA/CD LAN [1] or an 802.5 Token Ring [2,3].

Another serious issue with the high arrival rate of NOFs to a station on the ring is the potential for causing a ‘chain reaction’ of failures on the network due to overloading of that station, or in addition other stations on the network. For instance, as a result of severe congestion in the station that is receiving the NOFs, other processes in that station may not receive adequate computing resources. If the station is acting as a router, this may cause time critical algorithms in the router not to be executed in a timely fashion. The overhead for processing the frames delivered at such a high rate may cause buffers to overflow, thus increasing the probability of other time-critical traffic to be lost.

Another possible cause of a chain reaction may be as a result of the delivery of duplicates due to NOFs. When control messages for a network are delivered to a station such as a bridge or router, they are ‘aged’ out after some time based on their arrival time. To simplify processing of control messages, they are usually constructed with no sequence numbers in them. The assumption often made is that there is a very low probability of out-of-order or duplicate delivery. When a large number of duplicates are delivered because of circulating NOFs, old state information may continue to be maintained, when in fact the inherent aging mechanisms in the station would have otherwise removed them from the system. Thus, we believe that an explicit mechanism for the removal of NOFs is required.

There are several error conditions that may create NOFs. Some of these may be because of insertion/deinsertion of a station or merging of multiple rings or a ring configuration change without adequate consideration to the removal of frames on the ring. For example, a simplistic implementation of ‘graceful’ insertion/deinsertion can create NOFs [9]. These NOFs can result in NOFs with good CRC and, therefore, these frames may be forwarded over the Extended LAN or WAN.

Another source of errors that may create NOFs that we quantify here is due to ‘understripping’ of frames on the ring. The FDDI token ring MAC standard requires that an originating station be responsible for stripping frames it has transmitted on the ring. The originating station achieves this by stripping frames based on their source address matching the station’s ‘My Long Address’ (MLA). The algorithm is typically called the Source Address (SA) matching algorithm. In addition, bridges and other stations transmitting frames on behalf of other stations on an FDDI ring may use alternative algorithms, such as the Frame Content Independent Stripping (FCIS) algorithm, described in [17,18].

A frame is considered to be ‘understriped’ when the transmitting station on the ring fails to strip the frame from the ring after exactly one traversal around the ring. There are two forms of understripping that have different levels of severity in their effect over the entire network.

1. Frames understriped due to bit errors in the frames, which as a result may not have a good CRC. The effect of NOFs with CRC errors, while still serious to stations on the ring, are limited only to the local ring.

2. Frames understriped due to other error conditions on the ring, but these NOFs created have a good CRC. The effect of this second type of understripping and the resulting creation of NOFs is that they may be delivered not only to the stations on the local ring, but also propagated on the entire Extended LAN/WAN, also.

We address the first class of errors initially. The source address (SA) based frame stripping algorithm adopted for FDDI may understrip frames as a result of bit errors which alter the source address of the frame. We show in [17,18] that the probability of understripping with the SA based stripping, although small, is the most significant contributor to understripping of frames on the FDDI (and hence cause NOFs to be created). This probability for a typical ring [10,18] is:

\[
\text{Prob. (understripping with SA-match strip)} = 6.67 \times 10^{-10}.
\]

Thus, a NOF is created with a relatively high probability on the ring. On a token ring with an average load of 50% (say with approximately equal busy and idle intervals), with an average packet size of 512 bytes, we would have a frame understriped once every approximately 10,000 seconds (≈ 2.78 hours).

The second class of errors creating NOFs occur due to station failures, ring merging and the possibility of multiple tokens. Understripping due to the creation of a token due to bit errors may be quantified. For the ‘typical’ ring, the probability of creating a token is [17]:

\[
\text{Prob. (understripping by duplicate token creation)} = 1.25 \times 10^{-27}.
\]

The existence of multiple tokens can create understripping, for example when using an alternate stripping algorithm such as FCIS or other Non-SA Match algorithm.

### 2.2 Duplicate Tokens and Their Impact

When two or more tokens exist on the ring simultaneously, the ring is said to have a duplicate token problem. Since multiple tokens allows multiple stations to transmit simultaneously, these stations effectively collide into each other. This is the only known cause of collisions between transmitters in a token ring. Such a problem can occur due to bit errors, a duplicate address problem [16], merging of rings without proper scrubbing [4], or design that tends to make duplicate token more likely.

Duplicate tokens can increase frame loss on the ring, when multiple stations transmit simultaneously. Depending on the distribution of load, the period of multiple stations transmitting and colliding into each other can vary from a short period to con-
ceivably a very long period. This behavior can introduce non-
determinism on the FDDI token ring resulting in periods where
frame loss rate can approach 100%. This non-deterministic be-
havior can cause serious impact to real time control or time sen-
sitive applications (e.g., in a distributed real-time process control
application in which a deadline might be missed) [14]. Using a
pathological example, one can show that stations with symmetri-
cal load (e.g., infinite load case) can transmit and collide into
each other indefinitely when there is a duplicate token and when
the stations are perfectly synchronized with respect to the token
access. Therefore, during this period of orchestrated collisions
all of the frames transmitted may be lost. Although this behavior
results in a spectacular failure, the likelihood of occurrence in
real life is probably very low. It is more likely for the duplicate
token problem to cause transient or intermittent high rate of
frame loss. However, it has been known that transient and inter-
nitent faults in both computer systems and networks always
make it more difficult to isolate and recover [5].

As mentioned earlier, the duplicate token problem is also known
to cause understripping to stations that are using an alternative
frame stripping algorithms, which are Non-SA Match algorithm.
Frames understripped become NOFs. An alternative frame strip-
ing algorithm, such as the FCIS algorithm, uses other means
(such as a void frame) as a normal termination for stripping. As
an exception condition, most alternative frame stripping algo-

rithms unconditionally terminate stripping based on the recep-
tion of a token. Thus, the duplicate token can cause an alterna-
tive frame stripping algorithm to understrip. Given that the
duplicate token rate due to bit error is shown to be very low, the
understrip error rate due to this condition is also very low.

3 SOLUTIONS

In this section, we will present the set of Ring Purger algorithms
which are used to resolve the NOFs and the duplicate tokens
problems. While we need at least one station on the ring to par-
ticipate, and for the ring to benefit from this set of algorithms, it
is not required that all stations on the ring be capable of per-
forming these algorithms.

3.1 RING PURGER ELECTION ALGORITHM

The election algorithm is a distributed election algorithm which
is used to select one and only one station on the ring, called the
Ring Purger, to remove NOFs and duplicate tokens. The key
properities that the Ring Purger election algorithm must have are
the following:

- The algorithm ensures that there is one and only one Ring Pur-
ger on the ring most of the time.
- If the Ring Purger fails, the algorithm dynamically and auto-
matically selects a new Ring Purger with no disruption to the
ring operation. It provides timely detection of failure of the
current Ring Purger. Also, it minimizes the time for election of
a new Ring Purger.
- The algorithm elects a Ring Purger as quickly as possible, sub-
sequent to the ring becoming operational.

- The algorithm should avoid unnecessary switching of the Ring
Purger. Also, it should avoid premature election of an 'inap-
propriate' station as a Ring Purger.
- The algorithm must have no impact on the operation of the
ring - especially ring initializations.
- The algorithm uses a minimal amount of ring and system
bandwidth for the Ring Purger election related operations. Also,
it minimizes the burst load on the ring and receiving sta-
tions.
- The algorithm is robust to errors, such as bit error and conges-
tions. Also, it minimizes false detection of the failure of a
Ring Purger.

The election algorithm uses the ANSI Station Management
(SMT) Extended Service Frame (ESF) [4] to construct a Ring
Purger Hello frame and a Candidate Hello frame. These frames
are transmitted to a special Ring Purger Election multicast ad-

dress. All stations participating in this algorithm must receive
frames sent to the multicast address. The election algorithm uses
three criteria to select a Ring Purger: claim token winner, station
that was a Ring Purger prior to the ring initialization, and the
winner of the election process. Normally the ring should have a
single Ring Purger, but under exception conditions the ring may
have multiple Ring Purgers for a short period of time. The elec-
tion algorithm resolves a multiple Ring Purgers situation and se-
lects the Ring Purger with the highest address to remain. Any-
time during the operation of the ring, if the ring has no Ring
Purger then a distributed election process is started and a new
Ring Purger is elected.

Immediately after a ring initialization (i.e., a MAC state variable
RingOp changes from False to True), the station who was the
Ring Purger before the ring initialization and, in addition, the
station that won the claim token process become the Ring Pur-
ger. As soon as a station becomes a Ring Purger, the station
transmits a Ring Purger Hello frame and starts performing the
removal of NOFs and the detection and removal of duplicate to-

ks. In addition, anytime during the operation of the ring, sta-

ations.
tions which are not the Ring Purger expect to receive at least one Ring Purger Hello frame within a Purger Hello Receive time period. If a station fails to receive a Ring Purger Hello frame within the time period then the station also enters the election stage to see if it should become the Ring Purger. During the election stage, the station transmits a Candidate Hello frame periodically. If the station receives either a Candidate Hello frame with a higher SA or a Ring Purger Hello frame then the station ends its election stage. If the station remains in the election stage for an election time period, the station assumes that it has won the election and that the ring has no Ring Purger and, therefore, it becomes a Ring Purger. This is the third method for a station to become a Ring Purger. The election process ensures that either after a ring initialization or anytime during the operation of the ring, if there exists no Ring Purger then a new one is elected.

Any time during the operation of a station, a network manager can enable or disable the station’s Ring Purger function, without any disruption to the ring. Once a station’s Ring Purger function is enabled, the station begins to participate in the Ring Purger algorithms. The station quickly learns whether or not there is a Ring Purger in the ring. If the station receives a Ring Purger Hello frame then it knows that there is a Ring Purger. If there is no Ring Purger then the station enters the election stage. Also, a network manager can disable a station’s Ring Purger function at any time, without affecting the operation of the ring. If the station is currently a Ring Purger then the station ceases to be the Ring Purger by turning off its Ring Purger elections, ring purging and duplicate token detection algorithms. In this case, the ring will have no Ring Purger until the election process selects a new Ring Purger.

Anytime a station detects an internal fault, including a duplicate address problem, the station ceases its operations relating to the Ring Purger and its election. In order to enhance the stability of the ring, the Ring Purger monitors to see if the ring stays operational for a period of time after each ring initialization. If the Ring Purger detects the fact that the ring is continually unstable within a given time interval then the Ring Purger ceases its operation as a Ring Purger (i.e., it ceases to be a Ring Purger but it continues its other station functions). This prevents the Ring Purger from further complicating the ring instability situation where the ring is rapidly oscillating between ring operational state and ring initialization.

3.1.1 TIMING PARAMETERS FOR THE RING PURGER ELECTION

We use the key properties (outlined earlier) as guidelines for determining the values for the various timers related to the Ring Purger election algorithm. There are four timers involved in the entire Ring Purger election and operation. The first is the periodic announcement of the existence and operation of a station as a Ring Purger, which is achieved by the Ring Purger transmitting a Ring Purger Hello at a reasonable frequency. The time between Ring Purger Hellos is specified by the Ring Purger Hello Timer. The second timer is to detect the failure of a Ring Purger by the stations on the ring, which is the Purger Hello Receive Timer. The guiding principle relating these two timer values is that we want to minimize the false detection of a failure while meeting the above criteria. We know:

\[ P(\text{false detect Ring Purger failure}) = P(\text{loss of hello})^6 \times P(\text{loss of Ring Purger Hello}) \]

We want to keep the probability of false detection to be fairly small, while still allowing for loss of Ring Purger hellos. The ratio between the two timers is approximately the number of hellos we may allow to be lost before declaring the Ring Purger has failed. For a typical ring, with assumptions of severe loss of Ring Purger hellos (50%), we believe the following ratio is reasonable:


The second set of timers are the Candidate Hello Timer and the Election Timer. Using similar arguments for the ratio between these two timers, we believe a reasonable value of the ratio is:

Election Timer / Candidate Hello Timer = 6.

It is important that the impulse load at the stations during the election be minimized. When an election stage begins, all the stations participating in the election transmit a Candidate Hello on the expectation of the Candidate Hello Timer. A large number of stations may be involved in transmitting at a given time. If the stations were synchronized, say due to starting the timer on a common event, such as a ring initialization, then the receiving stations would receive a burst of hellos at the same time. This may cause some of the Candidate Hellos to be lost. It is, therefore, required that the Candidate Hello Timer interval be uniformly distributed over a reasonable range around the selected value to provide the necessary randomness in the transmission times of the Candidate Hellos from the different stations in the ring. The absolute value of the Election timer must be chosen so as to elect a new Ring Purger in a timely fashion.

3.2 RING PURGING ALGORITHM

Given that the impact of NOFs can be severe but the probability of NOF occurrence is low, the following properties are necessary for a good NOF removal algorithm.

- NOFs or long fragments are removed in less than or equal to one traversal.
- The algorithm has minimal impact to the ring operation (e.g., no disruption to ring operations).
- The algorithm has negligible effect on the usable ring bandwidth.
- Compliant to ANSI FDDI standards and interoperable with ANSI FDDI compliant implementations. This means the algorithm must interoperate with stations compliant to ANSI FDDI standard even though those stations may not have implemented this algorithm.
- The algorithm is simple, reliable, and robust.
- Within a single ring, the algorithm should allow more that one station to execute the algorithm simultaneously. This property allows the Ring Purger election algorithm to be optimistic.
The ring purging algorithm removes NOFs and long fragments from the ring transparently and with no disruption to the ring operation. One of the key design decisions was whether or not to detect the existence of NOF before invoking the removal of the NOF. In order to meet the desired properties of removing NOFs within one traversal on the ring and that the algorithm be simple, reliable and robust, it was decided to eliminate the NOF detection phase. Another important constraint is to minimize changes needed in the ANSI FDDI MAC standard. This is an important constraint because at the time this algorithm was being designed the ANSI FDDI MAC standard was already in its final stage of approval. It was considered to be of utmost importance to maintain stability and timely approval of the MAC standard and, therefore, the solution space was limited to the mechanisms and protocols allowed by the MAC standard. Given these considerations, the solution was to design an algorithm to automatically and transparently clean the ring whether or not there exist NOFs or long fragments.

In order to clean the ring of NOFs and fragments transparently without disruption to the ring operation, it was decided that the cleaning cycles must be synchronous with or be part of the token access opportunity. The algorithm uses the token to schedule the beginning of each purge cycle (also called the cleaning cycle). This decision was key to the purging algorithm as it allows a simple and elegant solution which can be easily added to the existing MAC protocol implementation. Given that there is no need for the NOF detection phase, the cleaning of the ring is initiated each time a token is received. In order to clean the ring transparently with negligible performance impact to the ring and stations, the purge cycle must not delay the token rotation. This means the algorithm should not hold on to the token while waiting to complete the purge cycle. So, the algorithm must release the token as soon as the purge cycle is started.

We now need a method to terminate the purge cycle. The purging algorithm uses a well-formed MyVoid frame, which is defined in the MAC standard, to mark the end of frames transmitted and the end of a purge cycle on reception. The MAC standard defines the well-formed MyVoid frame as a minimum size frame which can be transmitted by any station obeying the MAC protocol rules for frame transmission. In addition, the MAC standard specifies frame reception protocol rules which explicitly preclude the reception of any void frame.

Therefore, based on these considerations a void frame was chosen as the delimiter frame to mark the end of a purge cycle. The purging function is an integral part of the MAC sublayer functions and, therefore, its operation is tightly coupled with the rest of the MAC state machine. A Ring Purger performs purging only when the ring is operational. It ceases to purge the ring as soon as it detects that the ring is no longer operational.

Each time the Ring Purger receives a nonrestricted token or a restricted token it captures the token. If the Ring Purger has frames to transmit then the frames are transmitted following the MAC protocol rules for the type and amount of transmission. The Ring Purger transmits two MyVoid frames, to start a purge cycle, before transmitting the token. As shown in Figure 1, a MyVoid frame is a minimum size frame with the Destination Address (DA) and the Source Address (SA) set to the MAC's address (i.e., for FDDI it's the MLA). During a purge cycle, the Ring Purger unconditionally removes all frames and fragments received. When removing a frame or fragment, the Ring Purger replaces the entire frame or fragment with idles. Normally, a purge cycle is terminated by the reception of one of the error-free MyVoid frames. During a purge cycle if a MyVoid frame with error (e.g., FCS error) is received then the purge cycle is continued, which results in removing the MyVoid frame with error. In order to avoid the premature termination of the purge cycle, which can cause understripping of frames transmitted by the Ring Purger, the algorithm requires an error-free MyVoid to terminate the purge cycle. Also, in order to meet the requirement of low probability of overstripping, the algorithm requires the transmission of 2 MyVoid frames but allows 1 error-free MyVoid to terminate the purge cycle. The purge cycle is also unconditionally terminated when the Ring Purger receives a token, or a ring initialization frame (e.g., Claim Token or Beacon frame), or when the MAC determines that the ring is no longer operational.
two MyVoid frames before transmitting the token. It starts purging the ring by removing all frames and fragments received. While purging, it also looks for the reception of its MyVoid, token, or ring initialization frames (e.g., Claim Token or Beacon frame) to decide whether or not to terminate purging. Normally a purge cycle is terminated by the reception of an error-free MyVoid. If it receives a token in the middle of a purge cycle then it generates a Purge Error event to indicate an abnormal termination of a purge cycle. If it receives a Claim Token or Beacon frame then it exits purging and it performs appropriate actions specified in the MAC standard. While purging, it also looks for the reception of its MyVoid, token, or ring initialization frames (e.g., Claim Token or Beacon frame) to decide whether or not to terminate purging. Normally a purge cycle is terminated by the reception of an error-free MyVoid. If it receives a token in the middle of a purge cycle then it generates a Purge Error event to indicate an abnormal termination of a purge cycle. If it receives a Claim Token or Beacon frame then it exits purging and it performs appropriate actions specified in the MAC standard.

Figure 2 Ring Purging Flow Chart

The operation of the purging algorithm is shown in the spacetime diagram of Figure 3, where there are three stations on the ring and station S3 is the Ring Purger. Time increases from top to bottom and the direction of flow of token and frame is from left to right. Figure 3 shows several cases of the Ring Purger operation including the removal of NOF, its purge cycles on an idle ring, and its operation on a busy ring. The Ring Purger continuously purges the ring each time it receives a token. In Figure 3, station S1 transmits frame F1, after capturing the token. Later, station S1 fails to strip frame F1. Frame F1 becomes NOF1 and traverses a large part of the ring. Since S3 has no outstanding frames to transmit, it transmits two MyVoid frames and then the token immediately after receiving the token. It purges the ring until it receives one of its error-free MyVoid, thereby removing NOF1 transparently. The example shows that when S3 receives the token the second time, it has F2 to transmit. Therefore, it transmits F2 before starting another purge cycle. When the ring is idle, the Ring Purger continuously purges the ring each time it receives the token. As soon as the ring is busy, the ring is available to stations with frames to transmit (e.g., S1 and S2) and the purge cycle is reduced down to one purge cycle per token rotation (i.e., maximum token rotation time is 2 * TTRT).

3.2.1 Why Transmit 2 Void Frames?

The purging algorithm transmits two MyVoid frames as 'marker' to identify the trailing end of the transmission of its own frames. In order to be robust to the bit errors that may cause the marker to be lost, the purging algorithm transmits redundant void frames as the marker. If the marker were just a single void frame, and were to be lost due to bit errors, then the purging algorithm may result in one of two situations. In the case of the idle ring, the loss of the marker would result in a Purge Error being indicated on each purge cycle where the marker is lost. On the other hand, if the token ring is not idle (i.e., stations are transmitting frames during this purge cycle), the loss of the 'marker' would cause the Ring Purger to overstrip frames. Thus, more frames (other than the NOFs and frames transmitted by the Ring Purger station) would be incorrectly stripped by the Ring Purger. The termination of the frame stripping by the station would then occur only when either the token is encountered by the station or when the TVX expires, causing the ring to be initialized and thus recover from the over-
Assuming independent bit errors, the probability of the loss of a single void frame due to bit errors in a ‘typical’ ring is [17]:

\[
P(\text{loss of a void frame}) = 1.0 \times 10^{-5}
\]

We believe that this is too high a probability for a loss of the ‘marker’, potentially causing the Purge Errors to be very frequent. On a small ring which is idle, ignoring the overhead for token circulation, the probability of loss of the single void frame may be translated to having one lost every 0.224 seconds. To add robustness in the presence of such bit errors, we add a redundant void frame to be transmitted by the Ring Purger on each purge cycle. Thus, the Ring Purger would erroneously indicate a Purge Error (or overstrip in a loaded ring) if and only if both the void frames are lost. As shown earlier, the probability of losing both the void frames due to bit errors is \(1.0 \times 10^{-10}\). Thus, on an idle ring, the time between Purge Errors is a much more acceptable time period of 6.22 hours. On a ring with load, the time between Purge Errors and the probability of overstripping is even lower, since the number of opportunities to transmit void frames reduces with load.

3.3 THE DUPLICATE TOKEN DETECTION/REMOVAL ALGORITHM

As we have shown earlier, the impact of the duplicate token is undesirable although its occurrence may be a low probability event. The MAC standard has a simple and transparent method to remove the duplicate token if and when the duplicate token collides into a transmitter. If a station that is holding a token receives another token then the second token is removed and a duplicate token event is reported. In practice, this method is effective in removing many of the duplicate token occurrences. However, this method alone is not sufficient to ensure that the aforementioned impact of duplicate tokens is eliminated. Therefore, we need a more deterministic method to remove the duplicate token.

Instead of designing a separate mechanism to detect the existence of the duplicate token, we use the purging algorithm which continuously purges the ring. As we have described in the ring purging section, the Ring Purger generates a purge cycle each time a token is received. A purge cycle is normally terminated when the Ring Purger receives its own error-free MyVoid frame. When the ring has a duplicate token, the purge cycle will be terminated by the duplicate token. This is an important property which is used by the algorithm to detect the duplicate token problem. We use the termination of purge cycle to indicate if the purge cycle was terminated normally or terminated by a token. If a purge cycle is terminated by a token then we call it a Purge Error. This is described in Figure 2. The criteria for detecting a duplicate token can be summarized as the consecutive occurrence of a predefined number of Purge Error within a given time period (e.g., 10 seconds). If we assume that the Ring Purger’s MLA is not duplicated by another station’s address then this is an acceptable criteria to indicate the duplicate token problem. One way to ensure that the Ring Purger’s MLA is not duplicated is to ensure that the station’s duplicate address test does not indicate a failure when becoming a Ring Purger, and that the duplicate address test be run periodically. In addition, once a station’s duplicate address test indicates a failure, the station ceases its operations and participation in the Ring Purger algorithms, until the problem is resolved. Once the duplicate token detection criteria are met we can take the necessary actions to remove the duplicate token. A simple and acceptable method to remove the duplicate token is to force a ring initialization by initiation of a claim token process.

Given that the ring purging algorithm purges the ring each time a token is received, the occurrence of the duplicate token may force the ring to have two sets of purge cycles, where each purge cycle consists of 2 MyVoid frames and a token. This means that the first occurrence of a duplicate token will generate a Purge Error, but subsequent purge cycles will no longer generate Purge Errors as the ring has one purge cycle per token. Transmission of 2 MyVectors in conjunction with the duplicate token would result in each of the tokens on the ring being associated with 2 MyVectors. Thus, the Ring Purger would not detect a Purge Error subsequent to the first occurrence, and would fail to detect the duplicate token condition. The method to resolve this situation is simply to stop the purge cycle for at least two token rotations once a Purge Error is detected. This ensures that once a Purge Error is detected all the purge cycles are terminated and the ring is freed of all circulating MyVectors. After two token rotations then the purge cycles can be initiated again.

The duplicate token detection and removal algorithm performed by the Ring Purger can be summarized as follows. While the ring is operational (i.e., RingOp = True), the Ring Purger initiates a purge cycle each time a restricted or nonrestricted token is received, as part of its NOF removal. If a Purge Error is detected the Ring Purger stops initiating the purge cycle for at least two token rotations. After two token rotations, the Ring Purger starts initializing the purge cycle each time a token is received. If the number of consecutive Purge Errors is equal to or greater than a predefined limit within a predefined period of time then the Ring Purger removes the duplicate token by forcing a claim token process. Anytime the ring is initialized, the Ring Purger reinitializes the duplicate token detection and removal by resetting the test timer and the Purge Error count. For most FDDI ring operation, an acceptable criteria for duplicate token can be 4 consecutive Purge Errors within 10 seconds.

3.3.1 Reliability of Duplicate Token Detection

If the condition of a duplicate token persists over a long period of time, the performance of the ring may be severely degraded. Therefore, it is important to identify and attempt to take corrective action. When a purge cycle is terminated at the Ring Purger by a token, a Purge Error condition is logged. When a number of occurrences of the Purge Error, greater than a fixed threshold, are encountered over a time period of 10 seconds, the ring is initialized. This threshold, called the PurgeError Limit, is set to be 3. When the number of Purge Errors over a short time interval of 10 seconds is greater than the PurgeError Limit, the ring is determined to have a duplicate token. It is important that we do not cause false ring initializations however. The loss of void frames on the ring due to bit errors may result in the false detec-
tion of a Purge Error. For a false detection to occur and cause the ring to be initialized, we need to have a total of 4 pairs of 2 void frames to be lost during the time period of 10 seconds for the ring to be initialized. The probability of loss of a void frame, given the nominal bit error rate on the FDDI token ring of $2.5 \times 10^{-10}$, is given in [17] as follows:

Probability of loss of 2 Void frames = $10^{-10}$.

For a typical ring of 20 stations, with an extent of 1km, the token circulates every $25 \mu$s on an idle ring. Thus, in 10 seconds, we have 400,000 token opportunities to transmit 2 void frame each time on a typical idle ring. Thus, the probability of losing 4 void frame pairs of 2 each is:

Probability(losing 4 void frame pairs in 10 s) = $1.0667 \times 10^{-19}$.

This is the probability of false detection of Purge Errors that can cause the ring to be initialized due to the void frames being lost from bit errors. We believe that this is an extremely low probability event and observe that the Ring Purger algorithm is extremely robust to such false detection of Purge Errors. Thus the primary reason that > 3 Purge Errors occur within a period of 10 seconds would be because of a duplicate token condition, so that this would be a reliable means of detecting the condition.

4 PERFORMANCE IMPACT OF RING PURGING

Since 2 void frames are transmitted by the Ring Purger on every token rotation, it may give the false impression that there is a considerable performance impact of the ring purging algorithm on the token ring. We show in this section that it is not so and that in fact there is a negligible impact on performance on the ring as well as the stations due to the Ring Purger.

When the ring is idle, the Ring Purger has a token opportunity very frequently and, therefore, the Ring Purger transmits the void frames frequently. The rate of void frames seen on the idle ring is high. However, this has no performance impact on the ring, since the stations on the ring are not utilizing the time available for transmission on each token opportunity. The idle ring has primarily idles and the token circulating. The void frames only use up what would otherwise be wasted bandwidth of the ring. As shown in Figure 3, the idle ring will therefore only have the 2 voids and the token circulating on the ring. When a station does have a frame to transmit, the void frames transmitted by the Ring Purger trail these frames on the ring. Stations on the FDDI token ring do not receive void frames. Therefore, the high rate of circulation of voids on the ring do not have any impact on the performance of the stations.

The rate of voids transmitted by the Ring Purger on an idle ring depends on the ring latency $D$. When the ring latency $D$ is larger, fewer token opportunities are encountered by the Ring Purger in unit time (e.g., once per $D_{max} = 1.773$ msecs) and hence the transmission of voids is reduced. When the load on the ring increases from being idle, significantly fewer token opportunities are encountered by the Ring Purger (e.g., maybe once every $TTRT$, which can be once every 167.78 msecs) and thus the rate of voids per unit time reduces. The algorithm exhibits a desirable characteristic of reducing overhead with increasing load. The available bandwidth on the ring is not affected on the ring as the load increases from zero, almost until the point when the ring is loaded to its maximum. Only when the load approaches the maximum available bandwidth will the stations see a small reduction in the available bandwidth on the ring. We show below that the worst case loss in the maximum available bandwidth has an upper bound of 0.22%, which is negligible.

When the ring is loaded to nearly its full capacity, so that the total amount of transmission by all the stations on the ring for a token rotation is close to the Target Token Rotation Time ($TTRT$) value, the impact of the Ring Purger transmitting the two additional void frames will become visible. The Ring Purger will use up $5.08 \mu$s of the maximum available bandwidth on the ring for transmission of the two void frames on every token rotation. We may express the maximum available bandwidth in relative terms as a function of the values for $TTRT$ and $D$. Figure 4 shows the impact on the maximum available bandwidth on the ring with and without the Ring Purger, when the load is increased from zero (idle ring) to the maximum value. On a ring without a Ring Purger, the maximum available bandwidth (as shown in Figure 4) is equal to $TTRT \cdot D$. When the ring has a Ring Purger, the maximum available bandwidth is reduced to $TTRT \cdot D \cdot 2V$, where the value of $2V = 5.08 \mu$s. Therefore the reduction of the maximum available bandwidth due to the Ring Purger is not only negligibly small but occurs only when the stations on the ring have a total transmission load that is adequate to reach the maximum available bandwidth on that particular token rotation. The larger the maximum available bandwidth for a token rotation (larger value of $TTRT$), the smaller is the impact of the transmission of the 2 void frames. Thus the relative degradation in the maximum available bandwidth due to the Ring Purger gets to be smaller with larger values of $TTRT$.

![Figure 4 Performance Impact of Ring Purger](image_url)

Figure 5 shows the impact of the maximum available bandwidth on the ring for different values of $D$ as the value of $TTRT$ is varied from the minimum value allowed by the ANSI FDDI standard of 4 milliseconds to the maximum allowed value of 167.78 milliseconds. For very small values of $D$, the overhead is of the order of 0.1 to 0.2% for small values of $TTRT$ (of 4 msecs). However, as the value of $TTRT$ increases, the percentage impact on the maximum available bandwidth reduces very rapidly to
Percent Maximum negligible values. For larger values of $D$, as shown in Figure 5, the reduction in the maximum available bandwidth is even further reduced to extremely small values. In the worst case, with the minimum value of $TTRT$ and the value of $D \rightarrow 0$, the percentage loss in the maximum available bandwidth can reach a maximum value of 0.22%. For a typical ring [11], with 20 stations, and a 4 millisecond $TTRT$, the reduction in the maximum available bandwidth is of the order of 0.135%. If there happen to be multiple Ring Purgers active for a short interval of time (prior to the election algorithm disabling all but one of them), there is a slight increase in the overhead. Each Ring Purger will contribute to the reduction of the maximum available bandwidth by this small amount, on a heavily loaded ring. However, the election algorithm resolves this situation in a timely manner.

![Percentage Impact on Max. Usable B/W](image)

**Figure 5** Worst Case Ring Purger Overhead

Another minor impact on performance, is that the token is delayed by about 5 μs per token opportunity at the Ring Purger, which is again negligible for most applications. In fact, the ANSI FDDI MAC standard allows stations on the ring a maximum time ($L_{\text{max}}$) of 3.5 μs between frames. If stations did use this time budget allowed by the standard for most of the frame transmissions, their impact on the latency can be significantly larger than the impact of the total 5 μs latency contributed by the Ring Purger per token rotation. On the average, there is no effect on the usable bandwidth at idle or low loads or even at relatively heavy loads.

5 CONCLUSIONS

The NOFs and duplicate tokens are well-known problems for token rings. The IEEE 802.5 token ring has an explicit method to resolve the NOF and duplicate token problems. The FDDI token ring has implicit mechanisms that remove NOFs and detect and remove duplicate tokens effectively when they collide into a transmitting station. For all other cases of NOFs and duplicate tokens, the FDDI standard has not defined a solution. We provided an in depth analysis of the impacts of both the NOF and duplicate token problems. Although the probabilities of occurrence of NOF and duplicate token are quite low, their impact can be severe. The NOFs can cause serious congestion and waste of resources within stations on a single ring and, in addition, they can seriously impact stations and the network of Extended LANs or the WAN. The duplicate token can cause a high frame loss rate and can affect time sensitive applications where the high loss rate may cause critical deadlines to be missed. Therefore, a reliable method to resolve these problems is needed.

We presented a solution where a Ring Purger continually initiates a cleaning cycle to remove NOFs and to detect and remove duplicate tokens. A Ring Purger is a designated station on a ring that performs its station’s tasks on the ring and, in addition, performs the removal of NOFs and the detection and removal of duplicate token. A robust distributed election algorithm is used to select a Ring Purger on a ring transparently, without disruption to ring operations. Stations capable of performing the Ring Purger functions participate using the election algorithm to ensure that the ring has one and only one Ring Purger most of the time. There are three criteria for a station to become a Ring Purger in the following order: a station which won the claim token on the last ring initialization, station which was the Ring Purger prior to the ring initialization, and station which won the election process. The election process is a distributed n-party protocol where each participating station periodically transmits a Candidate Hello frame in a bid to elect the one with the highest address to become the Ring Purger at the end of the process. The Ring Purger also transmits a Ring Purger Hello frame periodically to notify all participating stations of the Ring Purger’s presence, and to resolve the multiple Ring Purger situation, if one exists. When multiple Ring Purgers exist on the same ring, then the one with the highest address will win the bid and remain as the Ring Purger. The election algorithm operates transparently to ensure that the ring has one and only one Ring Purger most of the time, and that the ring dynamically and automatically recovers from the failure of a Ring Purger by electing a new one.

The ring purging algorithm removes NOFs and long fragments transparently without disruption to the ring operation. Each time a restricted token or nonrestricted token is received, the Ring Purger initiates a purge cycle to clean the ring transparently of any NOFs or long fragments. Each time a token is received, the Ring Purger may transmit outstanding frames obeying the MAC protocol rules. Prior to transmitting the token, the Ring Purger transmits two MyVoid frames to start a purge cycle. It then transmits the token immediately. During a purge cycle, the Ring Purger removes all frames and fragments from the ring. It terminates the purge cycle if it receives an error-free MyVoid frame, a token, or a ring initialization frame (e.g., claim token or beacon frame). In addition, anytime the Ring Purger detects that the ring is no longer operational (i.e., $\text{RingOp} = \text{False}$) it terminates the purge cycle unconditionally. Given that the purge cycle cleans the ring transparently using the token to schedule its purge cycles, the purging algorithm works transparently and continuously to remove NOFs. Normally a purge cycle is termi-
nated by the reception of an error-free MyVoid frame. When a purge cycle is terminated by a token then the Ring Purger detects a Purge Error internally. If the Ring Purger detects a predefined number of consecutive Purge Errors within a specified time period then the Ring Purger determines that the ring may have a duplicate token condition. It then initiates a ring initialization by starting a claim token process to remove the duplicate token. Thus, the Ring Purger performs the removal of NOFs and long fragments, and it detects and removes duplicate tokens.

We have shown that the overhead due to the set of algorithms presented here is negligible. On an idle ring, the transmission of voids by the Ring Purger only uses up otherwise idle bandwidth. When the load on the ring goes up, the overhead due to voids reduces, since there are fewer token opportunities - a very attractive characteristic of the algorithm. Only when the load is close to the maximum available bandwidth do we encounter any impact on the performance. This is negligible, being about 0.135% of the maximum available bandwidth on a 'typical' ring, and the worst case reduction in the maximum available bandwidth is only 0.22%. When stations have higher bandwidth needs, or when the ring is larger, the relative overhead of the Ring Purger goes down quickly. Since stations do not receive voids, no system bandwidth is consumed for the Ring Purger operations. The election algorithm overhead is carefully controlled so that very little of the ring and system bandwidth is consumed for the Ring Purger election.

We have also shown that the set of algorithms used are robust to a large variety of error conditions. We have ensured that by transmitting 2 void frames as the marker by the Ring Purger, we add enough redundancy so as to avoid, almost completely, unnecessary or false ring initializations by the Ring Purger. It also reduces the probability of overstripping to an adequately small value. We showed that the duplicate token condition is reliably detected, by having an adequate threshold for the number of Purge Errors to occur over a small enough interval. It was shown that the probability of false detection of a duplicate token condition was extremely small. The algorithms are also robust to the duplicate address condition. We ensure that the Ring Purger does not continually cause the ring to oscillate. This is achieved by removing the eligibility of stations to be the Ring Purger when a duplicate address exists for the station or if the ring is experiencing frequent ring initializations. The timing parameters for election are also chosen so as to be robust to loss of election related frames due to bit errors and congestion.

In summary, the Ring Purger algorithms presented in this paper allow the FDDI token ring to continue to maintain high performance and correct operation even under the error conditions of No-Owner Frames and duplicate tokens. The algorithms are transparent to the operation of the ring, and are also robust to any of a variety of other error conditions on the ring.

6 REFERENCES

1. ANSI/IEEE Std. 802.3-1985, "Carrier Sense Multiple Access with Collision Detection (CSMA/CD)", Institute of Electrical and Electronic Engineers.


3. ANSI/IEEE Draft Appendix to ANSI/IEEE Std. 802.5-1989, "Enhancement for Multiple Ring Networks", (unapproved draft), P802.5/D15 (89/25), Institute of Electrical and Electronic Engineers.


15. Mamber, Udi, "Chain Reactions in Networks", IEEE Com-
puter, October, 1990.


7 ACKNOWLEDGEMENTS
We thank Tony Lauck and Bill Hawe for their support and encouragement during this work. We sincerely thank Paul Koning for his meticulous review and constructive inputs to this paper. We would like to thank Paul Clarfella for his dedicated work in implementing the algorithms described here and for his useful inputs. We would also like to thank the members of the Distributed Systems Architecture and Performance group for their feedback and suggestions.