AN EFFICIENT PRIORITY MECHANISM FOR RINGS

by

R. Cohen and A. Segall

Technical Report #606

January 1990
AN EFFICIENT PRIORITY MECHANISM FOR RINGS

Reuven Cohen & Adrian Segall
Dept. of Computer Science
Technion IIT
Haifa 32000, Israel

Abstract

The paper presents a new scheme for handling priorities in a token ring Local Area Network. The scheme of the IEEE-802.5 standard suffers from two principal drawbacks that are addressed in the present paper: it allows frame multiplication for a limited duration, and in some situations, a long time is required to achieve the desired priority level. This paper describes and proves a new priority mechanism that satisfies all properties of the standard, while effacing the abovementioned drawbacks.

1. INTRODUCTION

Access schemes in Local Area Networks (LAN's) are often required to allocate the capacity according to priorities. In a nonprioritized scheme, devices that have data to transmit, are given equal chance to do so. On the other hand, in a system that employs multiple priorities, the procedure must offer transmission opportunities in the correct order of priority [1].

Several capacity allocation techniques are currently in use in LAN networks, each corresponding to a specific Medium Access Control scheme [2]. The Timed-Token-Rotation scheme is used in the IEEE-802.4 token bus standard [3]. A station that captures the token can send data according to the value of some local clocks. The same principle is also used in the FDDI network [4]. Another scheme is Distributed Queuing, designed for the QPSX Metropolitan Area Networks [5]. In this scheme all stations share one distributed queue for each priority level.

In the present paper we consider the scheme of the IEEE-802.5 standard for token rings [1]. This scheme provides for 8 levels of priority, and uses two 3-bit fields, priority and reservation, in each frame.
This mechanism has two drawbacks that are addressed in the present paper:

- It allows frame multiplication in some situations for a limited duration. Because of this phenomenon, the IEEE priority mechanism cannot be applied to rings where no frame multiplication is allowed, like slotted rings [6]. Moreover, token supervision and recovery are greatly facilitated if exactly one frame exists on the ring at all times [7], [9].

- In some situations, a long time is required in order to achieve the desired priority level. Such situations may cause "starvation" and waist of bandwidth.

This paper describes a new priority mechanism that satisfies all properties of the standard, without requiring frame multiplication. Therefore, in addition to simplifying recovery in token rings, it can be directly implemented in slotted ring networks. Moreover, our protocol achieves the required priority level considerably faster than the standard protocol.

The organization of this paper is as follows: the next section describes the IEEE 802.5 standard approach to priority assignment, Section 3 introduces the new protocol and Section 4 presents the correctness proof.

2. PRIORITIES IN THE IEEE-802.5 STANDARD

The priority scheme is managed in the IEEE-802.5 standard [1] by means of two fields in the frame. The fields are designated \( P \) and \( R \) and they represent respectively the priority and the reservation. Figure 1 depicts the access control part of the frame format proposed by the standard. The fields \( P \) and \( R \) consist of 3 bits each, indicating 8 priority levels \([0, \ldots, 7]\). The field \( T \) is the token and \( M \) is the monitor bit, required for recovery purposes. The priority scheme can be summarized as follows:

- A station wishing to send a PDU (Protocol Data Unit) with priority \( P_m \) must wait for a free token \((T=0)\) with priority \( P < P_m \).

- A station may reserve a token with a desirable priority level \( P_m \) by setting the reservation field \((R)\) to \( P_m \), provided that the received \( R < P_m \).
When a station seizes a token, namely recognizes an empty token with $P \neq P_m$, it sets the token bit $T$ to full ($T \leftarrow 1$), resets the reservation field ($R \leftarrow 0$), and leaves the priority field unchanged.

A station that sends a PDU, retrieves the PDU upon its return to the station after a ring round trip. When a station retrieves its PDU, it issues a new empty token ($T \leftarrow 0$) and sets the priority $P$ as the maximum of $R$, $P$, and $P_m$. Thus, such a station may either increase the priority level or leave it unchanged, but cannot decrease it.

A station that increases the priority level is responsible for returning it to its former level. Such an action prevents situations where a token circulates the ring indefinitely just because its priority level $P$ is higher than all needed priorities.

Upon increasing the priority level, a station stores the values of the old and the new priorities. Later, when the station detects an empty token with a priority increased by itself, it decreases the priority. The new value is the maximum of the priority of the token before the station had increased it, the maximum reservation $R$, and $P_m$ - the maximum priority of locally waiting PDUs. This means that a station that increases the priority must remember the values before and after the increase. Also, if a station increases the priority from value $p$ to a value $p'$ say, and later decreases it to an intermediate value $p''$, it must remember $p$ and $p''$. This will enable it to later decrease the priority from $p''$ to $p$ or again to an intermediate value. Since a station may increase the priority level more than once before decreasing it, the abovementioned priority levels are kept in stacks. One stack $Sr$, is used for storing the priority level before the increase, and the
second stack, $S_x$, stores the new level. Values are pushed into the stacks every time the priority is increased by the station, and are popped when the priority is decreased to the old level. When a station decreases the priority level to an intermediate value, the latter replaces the top of $S_x$, while $S_r$ remains unchanged. This approach, where only a station that had increased the priority from $p'$ to $p''$ can decrease it from $p''$ to $p'$, in one or more steps, is the basic mechanism for achieving fairness. This approach ensures that if the priority is increased and then set to its previous value, the first station to benefit from the new level is the last that did so before the value had been increased. A station that increases the token priority is called a stacking station. It remains in this category as long as its stacks are not empty.

The above algorithm can be implemented with a minimum delay. Minimum delay in a token ring means that the value of a transmitted bit is either identical to the corresponding incoming bit, or a new value that depends on preceding bits and station state only. Therefore, the $i$'th bit can be transmitted by a station while it receives the $i$'th bit. The minimum delay is also called one-bit-delay, which is the time required to receive and transmit a bit without any further processing, since its value is unchanged or is determined in advance. Minimum delay at each station is important since it decreases the total ring delay and maximizes ring performance [7]. Moreover, longer processing delay leads to decreased reliability [8]. A protocol that allows recovery from arbitrary errors and failures in the ring and works with one-bit-delay is presented in [9].

There are some difficulties in achieving minimum delay at stacking stations. Consider a stacking station whose top of the $S_x$ stack holds a value equal to the priority field $P$. Such a station must decide according to the token bit whether to decrease the priority level. But the token bit comes after the priority field and clearly, determining the value of a field in the transmitted frame according to a later field in the received frame contradicts the minimum delay property. Moreover, testing any field, and then changing its value, even independently of other frame fields, cannot be done with one-bit-delay.

The IEEE standard copes with this difficulty in the following way: A stacking station transmits the value of $P$ unchanged, regardless of its incoming value. If it has a waiting PDU with priority equal to or greater than $P$, it sends the PDU if it recognizes an empty token. Both actions are performed independently of the contents of the stacks. On the other hand, if it does not have a waiting PDU with priority equal to or greater than $P$, it compares the value of $P$ with the top of its $S_x$ stack and does nothing if the values are...
different. If the two values are the same, the station must decrease the priority $P$ if the token is empty. Therefore, the station transmits a full token and tests the value of the incoming one. If it is full, namely the station is not supposed to alter $P$, it transmits the rest of the incoming frame with no change. On the other hand, if the value of the received token is empty, the station was supposed to change $P$ to the maximum of the top of $Sr$, the received reservation field and the priority of its most urgent waiting PDU. However, $P$ had already been transmitted by the time when the token is received. The problem is solved in the standard in the following way: the station changes the token bit to full in order to prevent other stations to use the token with the old priority. Then it generates a new frame with an empty token and with the appropriate priority. Now there are two frames in the rings and the first frame, with the full token and the old priority, is removed from the ring at the time when it is received back by the station.

The above procedure enables a stacking station to operate with one-bit-delay as long as the priority is not decreased, namely when: (1) the top value of its $Sx$ stack and the priority of the token are different, or (2) the station has a waiting PDU with a priority equal to or greater than the priority of the token, or (3) the token is full. When the stacking station must decrease the priority, it not only does not work with a one-bit-delay, but also temporarily generates a new frame.

In many cases frame multiplication, even temporary, may be unacceptable or at least undesirable, because:

- Token supervision and recovery are greatly facilitated if the access control protocol ensures that only one frame, either a free-token frame or a full-token frame, exists in the ring at all times.
- In slotted rings [6], a fixed number of frames, called slots, exist in the ring. Each station may use any empty slot to send its data, but the number of slots circulating in the ring must be fixed. This enables a station to recognize its slot and reset the token to empty when the frame returns after a round trip. Therefore, in slotted rings, frame or slot multiplication is not possible. Our new protocol eliminates this deficiency and achieves minimum delay with no need to multiply frames.

Another deficiency of the standard that is eliminated by our protocol refers to the time required to reach the desired priority level, especially when the gap between the token priority level $P$ and the required level $R$ is large. Consider an empty token with priority 7, the highest level. Suppose that the reservation field is 0, indicating that the priority field should be decreased to 0. Suppose also that there are seven
stacking stations, each having had increased the token by one level. In this case, the priority will be decreased in 7 steps: in the first step, the stacking station that had upgraded the priority from 6 to 7 will decrease it to 6; in the second step, the priority will be decreased from 6 to 5, and so on. The number of round trips required to achieve priority \( P = 0 \) depends on the relative location of the stacking stations, but in the worst case almost 7 round trips are required.

This phenomenon may cause:

i) Waste of bandwidth, since round trips are utilized for priority field changes rather than for data transmission.

ii) Starvation, since while the priority crawls to the desired level, an urgent PDU may arrive at some station. Such a PDU causes the reservation field to be increased, and the priority field may never reach the lower levels.

3. THE NEW PROTOCOL

3.1. Informal Description

Our new protocol is a modification of the IEEE standard, and has the following properties:

- It can be performed with one-bit-delay by every station (including stacking stations) except for a station that changes the token from full to empty. Note that such a station works with non-minimum delay in the standard of IEEE-802.5 as well. This fact has an insignificant contribution to the ring delay, since no more than one station can change the token from full to empty in any given round trip.

- Frame multiplication is not required.

- It reaches the required priority level considerably faster than in the standard, without unnecessarily going to intermediate levels.

Its basic elements are:
The frame format includes one priority field and two reservation fields. Note that two reservation fields are required, while in the standard one is sufficient.

Each station manages a local vector of 8 fields, instead of maintaining stacks. This vector plays the same role as the stacks in the standard, namely ensures fairness and liveness.

The rules for increasing the priority levels are the same as in the standard. The decreasing rules are different.

Figure 2 depicts the Access Control Part of the frame for the new protocol. It consists of five fields:

- $R_1$: The first reservation field (3 bits).
- $P$: The priority field (3 bits).
- $T$: The token field (1 bit).
- $M$: The monitor field (1 bit).
- $R_2$: The second reservation field (3 bits).

The token and monitor fields have one bit. The others have $\lceil \log_2 L \rceil$ bits, where $L$ is the number of priority levels.

It is easy to see that it is not possible to overcome the mentioned problems by simply changing the order of the fields in the standard. In fact, changing the order is not possible at all if one-bit delay is to be maintained: the reservation field must be located after the priority field, since a station should reserve a

---

<table>
<thead>
<tr>
<th>$R_1$</th>
<th>$P$</th>
<th>$T$</th>
<th>$M$</th>
<th>$R_2$</th>
</tr>
</thead>
</table>

FIGURE 2: Frame-Format

---
priority only if it cannot use the current one. Moreover, the priority field must be located before the token field, since the priority of the token must be known when receiving the token, in order to decide whether a full token should be transmitted instead of the received one.

In our proposal, the maximal reservation is held in the first reservation field $R_1$ when the token is empty, and in that case the value of $R_2$ is not used. On the other hand, when the token is full, $R_2$ holds the maximal reservation and $R_1$ is not used. A station that wishes to send data tries to enlarge $R_1$ to the value of the priority of its waiting PDU. Since when a station receives $R_1$ it does not know yet whether the token is empty or full, the reservation in $R_1$ is only tentative. If the station recognizes later that the token is full, it makes another reservation in the field $R_2$. We shall show presently that in this way a reservation can be made with a one-bit-delay.

After trying to enlarge $R_1$, a station that wishes to send data receives the priority field $P$ and tests whether this field is less than or equal to the priority of its waiting PDU. If it is, and the station does not change its value, the station is allowed to send its PDU provided that the token is empty. Therefore, the station sends a full token and then tests the received one: if it is also marked full, the station cannot send its data and makes a reservation in $R_2$. On the other hand, if the received token is empty, the station sends its PDU, and resets $R_2$ to 0. Since $R_2$ is always reset whenever a new PDU is sent, the frame received by the sender contains the maximal reservation, as determined by the ring stations in the last round trip. Note that in such a case, $R_1$ cannot be used to indicate the required reservation since its initial value is not 0, but rather equal to or greater than the priority field $P$ and as a result, $R_1$ may not contain the maximal reservation.

A station that sends a PDU enters ACTIVE mode. Upon receiving the frame back, it changes the token to empty and sets $P \leftarrow R_2$ if $R_2 > P$. This means that the station must wait for the incoming $R_2$ before transmitting $P$. Consequently, a station in ACTIVE mode does not operate with a one-bit-delay. If $R_2 < P$, namely the highest reservation is less than the current priority, there is only one station that may decrease the priority to $R_2$. This is the last station to have sent a PDU in a frame with priority less than or equal to $R_2$. This rule ensures fairness.

In our protocol, changes in the priority level are not remembered in stacks. Stacks are not convenient here, since if a station raises for example the priority from 0 to 2 and afterwards from 4 to 6, we
allow the station to decrease it in one step from 6 or 7 to 1 or 0. In the stacks structure, the station would have to search all entries in the stacks and not only the top ones in order to find out that it should reduce the priority. Instead, we use here a vector $V$ with 8 three-bit entries (assuming that there are 8 priority levels). The vector $V$ at station $i$ will be denoted by $V_i$ and its $r$'th entry by $V_i[r]$, where $r=0,1,2,\ldots,7$. Entry $V_i[r]$ indicates the value station $i$ should decrease the priority $P$ to, if it receives a frame with an empty token and fields $R_1$ and $P$ such that $R_1=r$, $P>r$ and $P>V_i[r]$. 

Our priority scheme can be summarized as follows:

a) A station can send a PDU with priority $P_m$ only when it receives a frame with an empty token, provided that it does not change the priority field of the frame and $P \leq P_m$ holds. As shown later, a station that receives a frame with an empty token and alters $P$ can only decrease it and allowing such a station to send its frame would result in an unfair scheme.

b) A station may reserve a token with a desirable priority level ($P_m$) by setting the reservation field $R_1$ to $P_m$, provided that the received $R_1 < P_m$. If the station detects later that the token is full, it sets the reservation field $R_2$ to $P_m$, provided that the received $R_2 < P_m$.

c) When a station seizes a token, namely recognizes an empty token with $P \leq P_m$ and does not change $P$, it sets the token bit $T$ to full and resets the reservation field $R_2$ ($R_2 \leftarrow 0$).

d) When a station retrieves its PDU, it issues a new empty token. The priority value is determined according to the reservation field $R_2$. If $R_2$ is larger than the received $P$, then $P \leftarrow R_2$. If they are equal, then $P$ does not change. If the received $P$ is larger than $R_2$, then $P$ either does not change, or is decreased to a value $\geq R_2$ if the conditions specified in e) are satisfied. In addition, the station performs $R_1 \leftarrow R_2$.

e) A station that increases the priority level from $p$ to $p'$, where $p'>p$, is responsible for decreasing it to any value in the range $[p,p'-1]$. The change is performed only if the token is empty and if the maximal reservation, held by $R_1$ when the token is empty, fulfills $R_1 < P$.

In order to illustrate rule e) above, suppose that $i$ increases $P$ from 0 to 3, and then $j$ increases it from 3 to 7. Then the vectors $V_i$ and $V_j$ are $V_i=(0,1,2,7,7,7,7,7)$ and $V_j=(3,3,3,3,4,5,6,7)$. This is because if the required value $R_1$ is 3, 4, 5 or 6, only $j$ can decrease $P$ to this value. On the other hand, if the required
value \( R_1 \) is 0, 1 or 2 then:

- if the frame is received by \( j \) and then by \( i \), the former reduces \( P \) to 3 and \( i \) reduces it to the desired value.

- if the frame is received by \( i \) first, then \( i \) reduces \( P \) directly to the desired value.

If \( P = 7 \) and \( R_1 = 5 \), then \( i \) cannot change \( P \), but \( j \) can decrease it to 6.

In order to understand the differences between our approach and the one of IEEE-802.5 consider the following scenario. Suppose that station \( i \) increases the priority from 0 to 2, then station \( j \) increases it from 2 to 4, and finally \( j \) increases it again from 4 to 6. Suppose also that after all PDUs with priority 6 are sent, the maximal reservation \( R \) is for priority 0. In the standard, at this time only \( j \) can decrease the priority, from 6 to 4. This is because \( k \) was the last to have increased \( P \) to 6. Then, \( j \) must again wait for the frame and decrease \( P \) from 4 to 2. When \( i \) recognizes that \( P = 2 \), \( T = \text{empty} \) and \( R = 0 \), it decreases \( P \) to 0. This process may take up to 3 round trips. In the worst case, 7 round trips are required: if \( i \) increases \( P \) from 0 to 1, and then to 2, 3, 4, 5, 6 and 7 and the reservation \( R \) is 0. In such a case, \( i \) must receive the frame 7 times in order to decrease \( P \) to 0. However, in our algorithm, when \( i \) receives the frame and recognizes that the maximal reservation is 0 and \( P > 0 \), it decreases the priority to 0 immediately. This is because \( i \) is the last station that had increased \( P \) from 0.

Next we describe the scheme by which a station can decrease the priority of an empty token with the minimum delay of one bit, without knowledge of the token contents. A station that receives the frame first tests the value of \( R_1 \) which represents the desired priority if the token is empty. Then, it checks its local vector \( V \). The entry \( V[r] \) represents the value to which the priority should be decreased provided that (a) the value in the reservation field \( R_1 \) is \( r \), (b) \( P > r \), (c) \( P > V[r] \) and (d) the token is empty. The station then sets \( P \) to be the minimum of \( V[R_1] \) and the received value of \( P \). As shown later, this can be done with a one-bit-delay. Also, as shown later, when the token is full holds \( R_1 \geq P \), and since \( V[R_1] \geq R_1 \), holds for every station (since no station decreases \( P \) to a value less than the maximal reservation), the value of \( P \) in fact does not change. On the other hand, if the token is empty, then \( R_1 \leq P \), so \( P \) will be set to \( V[R_1] \) if \( V[R_1] < P \). Therefore, the value of \( P \) is in fact decreased only when the token is empty, without explicit knowledge of the token contents.
To demonstrate the above process, consider the following example. Suppose that the local vector \( V_i \) of station \( i \) fulfills: \( V_i[3]=5 \). This means that if \( i \) recognizes a frame with an empty token, \( P > 5 \) and \( R_1=3 \), it should decrease \( P \) to 5. Suppose that \( i \) receives a frame with \( R_1=3 \) and \( P=7 \). According to the above rules, it should test \( T \) before it can decrease \( P \) to 3. But when it receives \( T \), it is too late to change \( P \). However, recall that \( R_1<P \) implies \( T=\text{empty} \), and since \( V_i[R_1]\geq R_1 \) always holds, \( i \) transmits as the new \( P \) the minimum of its incoming value, 7, and \( V_i[R_1] \), namely 5. If the new value is different from the old one, as it is in our example, then the token must be \( \text{empty} \). Otherwise, either \( T=\text{full} \) or \( T=\text{empty} \), but \( i \) should not decrease \( P \).

3.2. Formal Specification

Each station can be in one of two modes: In \textit{ACTIVE} mode, a station has owned access rights, namely has recognized a frame with an empty token and with \( P \) not larger than the priority of its most urgent waiting PDU, has transmitted its PDU, and is waiting to receive the frame back in order to remove it from the ring and to transmit a new frame with an empty token. The other mode is the \textit{LISTEN} mode.

Notations: A station, say \( i \), has the following variables:

- \( \text{mode}_i : \text{ACTIVE or LISTEN} \) as explained above. At initialization, \( \text{mode}_i=\text{LISTEN} \).
- \( P_{mi} : \) holds the priority of the most urgent waiting PDU at the station, where (-1) indicates that no PDU is waiting for transmission.
- \( V_i : \) a vector with 8 entries. \( V_i[r], r \in [0, 7] \) indicates the value \( P \) should be decreased to if (a) the maximal reservation \( R_1 \) is \( r \), (b) \( P>V_i[r] \), (c) \( P>R_1 \) and (d) the token is \( \text{empty} \). In particular, \( V_i[r]=7 \) if \( i \) is not allowed to decrease \( P \) when \( R_1=r \). At initialization, \( V_i[r]=7 \forall r \in [0, 7] \).

The global variables are the frame fields: \( R_1, P, T \) and \( R_2 \). At initialization \( R_1=P=0 \) and \( T=\text{empty} \). The value of \( R_2 \) at initialization is arbitrary. The fields \( R_1, P \) and \( R_2 \) are 3-bits long and are transmitted with the most significant bit first.

The algorithm at each station \( i \) is given in Table 1. The instructions in \textit{LISTEN} mode are performed in the given order while the station is receiving the frame. Instructions \( <O_1>-<O_2> \) and \( <A_2>-<A_3> \) refer to the local vector and can be performed off-line: after the station completes transmission of the frame...
header, but before it receives the frame for the next time. In \textit{ACTIVE} mode, a station does not work with minimum delay: it must receive the second reservation field \( R_2 \) before transmitting \( R_1 \).

We shall use superscript "-" to denote a value of a field in the received frame, and "+" to denote its value in the transmitted frame. E.g., \( R_1^- \) denotes the incoming value of \( R_1 \), and \( P^+ \) denotes the outgoing value of \( P \). These notations will be used in the specification of the algorithm, as well as in the correctness proof. In \textit{ACTIVE} mode, \( R_2 \) also has an intermediate value, that is determined after \( R_2^- \) is known, but can change before \( R_2^+ \) is transmitted. This intermediate value is denoted by \( k_2 \).

Note that in IEEE-802.5 there is a distinction between the \textit{token-frame} and the \textit{data-frame} formats. However, in slotted rings such a distinction does not exist, and the only difference between a \textit{token-frame} and a \textit{data-frame} is that in the first the token is empty and therefore all stations disregard the data field, whereas in the latter the token is full. In our algorithm both versions are possible. Therefore, a station in \textit{ACTIVE} mode that receives its transmitted frame back, may either release a \textit{token-frame} that contains only the fields \( R_1, P, T \) and \( R_2 \), or keep the frame structure unchanged while updating those fields.
Table 1: The algorithm at station $i$ while receiving a frame header

If $mode_i = \text{LISTEN}$ then:

$L_1$: $R^i_t \leftarrow \max(R^i_0, P_{mj})$
$L_2$: $P^i \leftarrow \min(V_i[R^i_0], P^i)$
$L_3$: if $P^i \leq P_{mj}$, then \( P^i = P^i \land T^i = \text{empty}, \text{then}\) \( (T^i \leftarrow \text{full}, R^i_t \leftarrow 0, \text{send the PDU}, \ mode_i \leftarrow \text{ACTIVE}) \)
$L_4$: else $R^i_t \leftarrow \max(R^i_t, P_{mj})$
$L_5$: perform $<O_1>$ and $<O_2>$

If $mode_i = \text{ACTIVE}$ then:

$A_1$: $R^i_t \leftarrow \max(R^i_2, P_{mj})$
$A_2$: $T^i \leftarrow \text{empty}$
$A_3$: if $R^i_t \geq P^i$ then \( (P^i \leftarrow R^i_t, R^i_t \leftarrow R^i_t, R^i_t \leftarrow 0) \)
$A_4$: \( \forall r \in [0, P^i - 1], \text{do } V_i[r] \leftarrow \min(V_i[r], P^i) \)
$A_5$: \( \forall r \in [P^i - 1, \infty), \text{do } V_i[r] \leftarrow r \)
$A_6$: \( \text{if } \forall r \in [0, P^i - 1], \text{do } V_i[r] \leftarrow r \}
$A_7$: $mode_i \leftarrow \text{LISTEN}$

$O_1$: \( \forall r \in [P^i, \infty), \text{do } V_i[r] \leftarrow 7 \)
$O_2$: \( r \leftarrow P^i - 1; \text{while } (r \geq 0) \text{ and } (V_i[r] \neq r), \text{do } (V_i[r] \leftarrow 7, r \leftarrow r - 1) \)
Before presenting the correctness proof for the protocol we shall illustrate its operation on the ring of Fig. 3. At initialization, the mode of all stations is \textit{LISTEN}, and the frame header is: $(R_1, P, T, R_2) = (0, 0, \text{empty}, 0)$. The vector $V$ at each node is $(7, 7, 7, 7, 7, 7, 7, 1)$.

Suppose that the first node to receive the frame is $i$, and that it has no PDU waiting for transmission. Therefore, $i$ transmits $R_1^+ = \max(0, -1)$, $P^+ = \min(7, 0)$, $T^+ = \text{empty}$ and $R_2^+ = \max(0, -1)$. Thus, all frame fields, as well as $V_i$, the local vector of $i$, do not change.

Suppose that $j$ has a waiting PDU with priority 2. Therefore, it transmits $R_1^+ = \max(0, 2)$, $P^+ = \min(7, 0)$, $T^+ = \text{full}$ and $R_2^+ = 0$. Then it transmits its waiting PDU. However, $V_j$ does not change. The frame is then received by $k$, where $P_{m_k} = 4$. Therefore, $k$ transmits $R_1^+ = \max(2, 4) = 4$, $P^+ = \min(7, 0) = 0$, $T^+ = \text{full}$, $R_2^+ = \max(0, 4) = 4$, the frame still contains the PDU of $j$ and $V_k$ does not change. The frame is then received by $l$, where $P_{m_l} = 4$. Therefore, $l$ transmits $R_1^+ = \max(4, 4) = 4$, $P^+ = \min(7, 0) = 0$, $T^+ = \text{full}$ and $R_2^+ = \max(4, 4) = 4$, namely, none of the frame fields changes. The local vector of $l$ does not change either. Station $i$ still has no waiting PDU, and transmits the frame unchanged. Its local vector also remains unchanged. When $j$ receives the frame its mode is \textit{ACTIVE}. Therefore it executes the algorithm of an \textit{ACTIVE} station. Suppose that $P_{m_j} = 2$. Consequently, $R_2^+ = \max(4, 2)$ and $T^+ = \text{empty}$ hold following the execution of $<A_1>$ and $<A_2>$. Since $R_2 = 4$ and $P^+ = 0$, station $j$ executes $<A_3>$--$<A_5>$. Consequently $P^+ = 0$, $R_1^+ = 4$, $R_2^+ = 0$.

\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{figure3}
\caption{Ring Example}
\end{figure}
Thus, the frame transmitted by \( j \) now has: \((R_{1}, P, T, R_{2})=(4, 4, \text{empty}, 0)\). While it executes \(<A_{4}>-<A_{5}>\), station \( j \) changes its local vector \( V \) to \((0, 1, 2, 3, 7, 7, 7)\). Following the execution, \( j \) enters \textit{LISTEN} mode. Note that \( P=4 \) now, the maximal reservation from the previous round trip. Therefore, only stations holding \textit{PDU} with priority 4 or higher will be able to transmit now.

Suppose that the next station, \( k \), has \( P_{mi}=4 \). Therefore, following the execution of the algorithm, \( k \) enters \textit{ACTIVE} mode and the frame holds the \textit{PDU} of station \( k \) with header \((R_{1}, P, T, R_{2})=(4, 4, \text{full}, 0)\). The local vector of \( k \) does not change. Suppose that the next station, \( l \), has \( P_{mi}=7 \), in which case \( l \) transmits the frame with \( R_{1}=7, P=4, T=\text{full} \) and \( R_{2}=7 \). The frame still contains the \textit{PDU} of station \( k \) and \( V_{i} \) does not change. The next to act on the frame is \( k \), which enters \textit{LISTEN} mode and transmits the frame with \((R_{1}, P, T, R_{2})=(7, 7, \text{empty}, 0)\), while \( V_{k} \) is \((4, 4, 4, 4, 4, 4, 4, 7)\).

Suppose that now \( l, i, j \) and \( k \) have waiting \textit{PDUs} with priority 7, in which case the priority field \( P \) remains 7 until after all these \textit{PDUs} are transmitted. The stations send these \textit{PDUs} in a round robin manner: the first station will be \( l \), then \( i \) and so on, as long as there are \textit{PDUs} with priority 7 in some station. Suppose that the last station to send a \textit{PDU} with priority 7 is \( k \), and that all stations of interest have made a reservation for priority 3 in the frame containing the \textit{PDU} of \( k \). Therefore, when \( k \) receives the frame, the frame header is: \((R_{1}, P, T, R_{2})=(7, 7, \text{full}, 3)\). Note that \( R_{1}=7 \) because of the last reservation for such a \textit{PDU}. However, \( R_{1} \) is irrelevant when \( T=\text{full} \), and \( k \) looks only in \( R_{2} \) for the maximal reservation. Since \( R_{2}=3 \) and \( P=7 \), station \( k \) executes steps \(<A_{4}>-<A_{5}>\). Consequently, \( R_{1} \leftarrow 3, R_{2} \leftarrow 0 \) and \( P \leftarrow \min(V_{k}[3], 7)=4 \). Note that \( k \) cannot decrease \( P \) to the required value 3, since from the fairness considerations, this can be done only by the last station to have sent a \textit{PDU} in a frame with priority \( P \leq 3 \), namely by \( j \). However, since \( k \) is the last to have sent a \textit{PDU} in a frame with priority \( P \leq 4 \), it can decrease the priority to this intermediate value. Following the execution, \( V_{k} \) has 7 in all its entries. Station \( l \), the next to receive the frame, has \( P_{mi}=3 \). However, at \( i \) holds \( P^{*}=4 \), so \( l \) cannot send its waiting \textit{PDU}. This will be the situation also with the next station, \( i \). When \( j \) receives the frame, it has \( P_{mi}=3 \). Since \( V_{j}[3]=3 \), station \( j \) sets \( P^{*}=3 \) in step \(<L_{2}>\), namely \( P \) gets the desired value. However, \( j \) cannot send its \textit{PDU} since \( P^{*}=4 \), so one of the requirements in \(<L_{2}> \) is not satisfied. This ensures fairness, since \( j \), the last station to have sent a \textit{PDU} in a frame with priority \( P \leq 3 \), has previously increased the priority and blocked its downstream neighbor, \( k \), from doing the same. Therefore, the first to be allowed to send a \textit{PDU} with priority \( P \leq 3 \) should be the downstream neigh-
bor of \( j \). The local vector of \( j \) is now \((0,1,2,7,7,7,7,7)\), so that if the priority needs later to be decreased to 0, 1 or 2, the station that will do so will be \( j \), the last station to have sent a PDU with priority \( P \leq 2 \).

4. CORRECTNESS PROOF

In this section, we prove the following:

1. The new algorithm can be performed with one-bit-delay by a station in \textit{LISTEN} mode without multiplying tokens (this property does not exist in the standard of IEEE-802.5).

2. It ensures liveness, namely:
   (a) PDUs with higher priority are always transmitted first.
   (b) Within at most 1 round trip after the token becomes empty, either one of the stations transmits its PDU or the priority field \( P \) receives the value of the maximal reservation.
   (c) The token cannot be empty for more than 2 consecutive round trips if there are waiting PDUs somewhere in the ring.

3. It ensures fairness, namely, all stations that hold PDUs with the same priority have equal access rights to the ring.

The proofs of the above properties appear in Theorems 1, 2 and 3 respectively.

**THEOREM 1 (minimum delay)**

The algorithm in \textit{LISTEN} mode can be implemented with a one-bit-delay.

Proof:

We only have to consider \(<L_1>, <L_4>\), since \(<L_3>\) invokes \(<O_1>\) and \(<O_2>\) and the latter can be performed off-line, namely after the transmission of the frame header.

To prove the property for \(<L_1>, <L_2>\) and \(<L_4>\), we need to show that a station can transmit with one-bit-delay some field \( X^* \) whose value is the maximum/minimum of its incoming value \( X^- \) and some known value \( Y \). Recall that in \( R_1 \), \( P \) and \( R_2 \), the most significant bit is received first. Let \( X^-=(X(1)^-,X(2)^-,X(3)^-) \), \( X^+=(X(1)^+,X(2)^+,X(3)^+) \) and \( Y=(Y(1),Y(2),Y(3)) \). In order to transmit
A station should do the following:

1. If \( Y(k) = 1 \), transmit \( X(k)+ \)

2. If \( X(k)+ < X(k) \) and \( Y(k) = 0 \), transmit \( X(k)+ \)

3. If \( X(k)+ = Y(k) \), then transmit \( X(k)+ \)

4. Otherwise, if \( k = k+1 \) and \( k < 3 \), transmit \( X(k)+ \)

Note that in the IEEE-802.5 standard, the algorithm in ACTIVE mode cannot be performed with a minimum delay. The delay it contributes to each round trip is not substantial however, since only one station may be in ACTIVE mode in each round trip, as shown in Lemma 1 below.

**Lemma 1**

Every station that receives an empty token is in LISTEN mode. Every station that receives a full token is in LISTEN mode, except for the last station that has changed the token from empty to full.

**Proof:**

Suppose that the statement of the Lemma is not true. Since at initialization the token is empty and all stations are in LISTEN mode, the Lemma holds at that time. Let \( t \) be the first time when one of the two parts of the lemma does not hold. Namely, time \( t \) is the first time when one of the following happens:
• Some station $i$ receives an empty token in \textit{ACTIVE} mode.

• Some station $i$ receives a full token in \textit{ACTIVE} mode, but it is not the last station that has previously changed the token from empty to full.

Let $t_1$ denote one round trip before $t$. Since at time $t_1$ station $i$ enters the \textit{ACTIVE} mode, then from $<A_3>$ follows that it receives an empty token and transmits a full one. Hence station $i$ has changed the token from empty to full one round trip before $t$, so that both situations above require that there is a time $t_2$, between $t_1$ and $t$, when another station, $j$ say, changes the token from full to empty. This can happen only if station $j$ executes at $t_2$ the algorithm of the \textit{ACTIVE} mode. However, at time $t_2$ the statement of the Lemma is contradicted, since $j$ receives a full token in \textit{ACTIVE} mode, but it is not the last station that has changed the token to full. This contradicts the assumption that the first event that contradicts the Lemma takes place at time $t$.

\textbf{Lemma 2}

For every station $i$ holds:

(a) $V_i[r] \geq r$, $\forall r \in [0,7]$

(b) $V_i[r] \leq V_i[r+1]$, $\forall r \in [0,6]$

(c) $V_i[r] = r'$ implies $V_i[r'] = r'$ for $r' > r$.

\textbf{Proof:}

To prove (a) by induction note that it holds at initialization, since at that time $V_i[r] = 7$, $\forall r \in [0,7]$. An entry $V_i[r]$ can change in $<A_4>$, $<A_5>$, $<O_1>$ and $<O_2>$. However, in $<A_5>$ the entry $V_i[r]$ receives the value $r$, in $<O_1>$ and $<O_2>$ it receives the value 7 and in $<A_4>$ it can only increase, so the claim holds.

In order to prove (b) and (c), note that after proving (a) it is sufficient to prove that $V_i[r] \leq V_i[r+1]$, $\forall r \in [0,6]$ with equality if $V_i[r] = r$.

We prove this by induction as well. It will be convenient to use the superscript ".-" and "+" to denote the values of each entry before and after the execution respectively. At initialization $V_i[r] = 7$, $\forall r \in [0,7]$, so this claim holds. Consider now the events that can change $V_i$. 

Technion - Computer Science Department - Technical Report CS0606 - 1990
• If \( i \) performs \(<O_1>\) and \(<O_2>\), then from some \( r' \) and up, all entries \( V_i[r'], \ldots, V_i[7] \) receive the value 7. Therefore, the claim clearly holds for every \( r \geq r' \). The claim also holds for every \( r < r' - 1 \) according to the induction assumption since \( V_i[1], \ldots, V_i[r'-1] \) are not changed. The value \( r' \) is the first one encountered when descending from \( r = 7 \) for which \( V_i[r'-1] = r'-1 \). From the fact that \( V_i[r'] \) changes to 7, follows \( V_i[r'-1] \leq V_i[r'] \), so the claim holds also for \( r = r' - 1 \).

• If \( i \) performs steps \(<A_4>\) and \(<A_5>\):

  - For every \( r \) such that \( 0 \leq r \leq r'-2 \), if \( V_i[r] \) changes to \( P^- \), then \( V_i[r+1] > P^- \). However, from the induction assumption follows that \( V_i[r] \leq V_i[r+1] \). Therefore, \( V_i[r+1] > P^- \) holds too, so \( V_i[r+1] \) also changes to \( P^- \) and the claim holds. If \( V_i[r] \) does not change and \( V_i[r'] \geq r \), then from the induction assumption follows that \( V_i[r'] = V_i[r+1] \). Therefore, \( V_i[r+1] \) does not change either, and we have \( r \leq V_i[r'] = V_i[r+1] \), so the claim holds. If \( V_i[r] \) does not change and \( V_i[r'] = r \), then from the induction assumption follows that \( V_i[r] \leq V_i[r+1] \); and whether \( V_i[r+1] \) changes to \( P^- \) or not, this inequality holds also after the execution since \( P^+ > r = V_i[r'] \).

  - For \( r = r'-1 \) if \( V_i[r+1] = P^- \), then the claim holds since from \(<A_1>\) we have \( V_i[P^-] = P^- \). If \( V_i[r+1] < P^- \), then from (a) above follows that \( V_i[r] = r = P^- - 1 \) and the claim holds since \( V_i[P^-] = P^- \).

  - For every \( r \) such that \( r - 1 \leq r \leq r' + 1 \), the claim clearly holds since \( V_i[r] = r \) and \( V_i[r+1] = r + 1 \).

  - For \( r = r' + 1 \) holds \( V_i[r+1] > P^- \), while \( V_i[r+1] > r + 1 \) holds from (a), so the claim holds.

  - For every \( r \) such that \( P^- > r \leq P^+ \), \( V_i[r] \) and \( V_i[r+1] \) do not change, and the claim follows from the induction assumption.

\[ \square \]

Lemma 3

When a station \( i \) receives the frame, holds: if \( T = \text{empty} \), then \( R_i = P^- \) and if \( T = \text{full} \), then \( R_i \geq P^- \).

Proof:

By induction. At initialization, \( R = P = 0 \) and the token is \( \text{empty} \). Therefore, this claim holds for the first station that receives the frame. Suppose that the claim holds when some station \( i \) receives the frame. Now we shall prove that the Lemma holds also when the next station receives the frame, namely, \( T = \text{empty} \) implies \( R_i \leq P^- \), and \( T = \text{full} \) implies \( R_i \geq P^- \), where the + refers to the frame fields transmitted by station \( i \).
If \( i \) is in \texttt{ACTIVE} mode, if \(<A_3>\) is executed, then both \( R_1 \) and \( P \) get the value of \( R_2 \), so we have \( P^* = R_1 \), and the claim holds. On the other hand, if \(<A_4>\) is not executed, then \(<A_8>\) and \(<A_7>\) are performed. In \(<A_8>\), the field \( R_1 \) is set to \( R_2 \), which is not larger than \( P^- \), so \( P^- > R_1 \). If following \(<A_7>\) holds \( P^* = P^- \), then we have \( P^* > R_1 \), and the claim holds. Otherwise, namely if \( P^* < V_i[R_1] \), then from Lemma 2(a) holds \( V_i[R_1] > R_1 \), so \( P^* > R_1 \), and the claim holds too.

If \( i \) is in \texttt{LISTEN} mode and \( T^- = \text{full} \), then according to the induction assumption \( R_1 \geq P^- \). Following the execution of step \(<L_1>\) holds \( R_1 \geq P^- \), and after step \(<L_2>\), we have \( P^- > P^+ \). Therefore, \( R_1 \geq P^+ \), and since \( T^- = \text{full} \), the claim holds. If \( i \) is in \texttt{LISTEN} mode and \( T^- = \text{empty} \) then according to the induction assumption \( R_1 < P^- \). Consider now the following cases:

- If \( R_1 = P^+ \), then the claim holds regardless of the value of \( T^- \).
- If \( R_1 < P^+ \), then the induction assumption \( R_1 \leq P^- \) and \(<L_1>\) imply that \( P_m < P^+ \). Therefore, \( T^- = \text{empty} \) and the claim holds.
- If \( R_1 > P^+ \), we want to show that in addition to \( T^- = \text{empty} \), also holds \( P^* = P^- \) and \( P_m \geq P^+ \), so \( T^- = \text{full} \). First note from \(<L_1>\) that \( P^* = P^- \), since otherwise \( P^* = V_i[R_1] \geq R_1 \) (where the inequality follows from Lemma 2(a)), contradicting \( R_1 > P^+ \). Moreover, since we have \( R_1 \leq P^- \), \( R_1 > P^+ \) and \( P^* = P^- \), then \( R_1 < R_2 \). Therefore \( R_1 = P_m \) and hence \( P_m > P^+ \). Therefore, all conditions in \(<L_3>\) hold, resulting in \( T^- = \text{full} \). \( \square \)

Lemma 4

A station in \texttt{LISTEN} mode:

(a) does not change the priority of a \texttt{full} token.

(b) does not increase the priority of a token.

(c) does not decrease the value of \( R_1 \).

Proof:

(a) We have \( V_i[R_1] \geq R_1 \geq P^- \), where the inequalities follow from Lemma 2(a), \(<L_1>\) and Lemma 3 respectively. Hence following the execution of \(<L_2>\) holds \( P^* = \min(V_i[R_1], P^-) = P^- \).
(b) The only way to change \( P \) in \textit{listen} mode is to decrease it in step \( <L_2> \).

(c) The only way to change \( R_1 \) in \textit{listen} mode is to increase it in step \( <L_1> \).

\[ \square \]

\textbf{Lemma 5}

Consider the situation after some station \( i \) has transmitted the frame (with priority \( P^+ \)) and has changed its local vector \( V_i \) according to the algorithm. For every \( r \in [0, P^+ - 1] \), there exists exactly one station in the ring that will receive the frame in the next round trip while \( V[r] = r \) (note that one of those stations can be \( i \) itself).

\textbf{Proof:}

We start by pointing out two basic facts that will be used in the proof:

(a) In \( <O_1> \) and \( <O_2> \), in the range \( r \in [0, P^+ - 1] \), only entries with \( V[r] = r \) are altered.

(b) \( P \) can be increased by a station \( i \) only in \( <A_3> \).

(i) At that time, every other station \( j \neq i \) has \( V_j[r] = 7 \) for all \( r \) in the range \( r \in [P^+, 7] \); this is because station \( i \) is in \textit{active} mode and thus all other stations have been in \textit{listen} mode when they received the frame for the last time beforehand (Lemma 1), have left the priority unchanged (Lemma 4(a)), and have performed \( <A_4> \) with this value of the priority.

When \( <A_4> <A_5> \) are performed, then:

(ii) in the range \( r \in [0, P^- - 1] \), entries with \( V_i[r] = r \) are not altered, since \( V_i[r] = r < P^+ \).

(iii) in the range \( r \in [0, P^- - 1] \), no new entry with \( V_i[r] = r \) can appear, since every entry that is changed becomes \( V_i[r] = P > r \).

(iv) all entries in the range \( r \in [P^-, P^+ - 1] \) become \( V_i[r] = r \).

Now we return to the proof of the Lemma. Since local vectors can be changed only when receiving a frame, it is sufficient to prove that at the time after \( i \) has transmitted the frame, there exists exactly one station in the ring with \( V[r] = r \) for every \( r \in [0, P^+ - 1] \). The proof proceeds by induction. At initialization, the frame has \( P = 0 \), so the Lemma holds trivially. Suppose that the Lemma holds until some station \( i \) receives the frame and changes either \( P \) or its local vector \( V_i \);
Lemma 6

(a) When a station in ACTIVE mode receives the frame, \( R_2 \) contains the value of the maximal \( P_m \) from the previous round trip.

(b) A frame with an empty token contains in its \( R_1 \) field the maximal \( P_m \) from the previous round trip.

Proof:

(a) When a station enters ACTIVE mode and changes the token from empty to full, it sets \( R_2 \leftarrow 0 \) (step \(<L_2>\)). While the frame circulates in the ring, any station with a waiting PDU sets \( R_2 \) as the maximum of the received value of \( R_2 \) and the priority \( P_m \) of the waiting PDU.

(b) A token can be changed from full to empty only by a station in ACTIVE mode. At that time, the value of \( R_2 \), which from (a) carries the maximal reservation of the last round trip, is copied into \( R_1 \). Later, as long as the token is empty, the value of \( R_1 \) may be increased only by stations whose waiting PDU has higher priority. Therefore, when the token is empty, \( R_1 \) holds the maximal \( P_m \) since the last time when the token became full. Since station priority \( P_m \) does not decrease unless it transmits a frame, the value of \( R_1 \) is also the maximal \( P_m \) from the most recent round trip.

Lemma 7

Suppose that some station \( i \) transmits an empty token. At the time when it transmits the frame for the next time holds \( R_i \geq P^* \).

Proof:
Suppose that the statement is incorrect, namely \( i \) transmits an empty token, with arbitrary \( P^* = p \) and \( R^* = r \), and one round trip later transmits the frame with \( P^* = p' \) and \( R^* = r' \), where \( p' > r' \). From Lemma 3 follows that in the second transmission of \( i \) holds \( T^* = \text{empty} \) and therefore during the entire round trip between the two transmissions of \( i \), all stations receive and transmit an empty token (and hence are, and stay, in LISTEN mode). By Lemma 4, none of these stations can increase the priority, hence

\[
p \geq p' > r'
\]

This says that \( r' < p \), thus Lemma 5 implies that exactly one of those stations, \( j \) say, receives the frame when \( V_j(r') = r' \). Let \( p'' \) and \( r'' \) be the values of \( P^* \) and \( R^* \) transmitted by \( j \). Then we have:

\[
p'' > p' > r''
\]

where the first inequality follows from Lemma 4(b), the second from \( <L_2> \) in the algorithm of \( j \), and the third holds due to \( r'' < r' \) (Lemma 4(c)) combined with the monotonicity of the vector \( V \) (Lemma 2(b)). However, this last equation contradicts the fact that \( p' > r' \).

THEOREM 2 (liveness)

(a) If a station \( i \) gets access rights and transmits a PDU with priority \( P_m \), then no other station had in the previous round trip a waiting PDU with priority higher than \( P_m \).

(b) Within at most 1 round trip after the token becomes empty, either one of the stations transmits its PDU or the priority field \( P \) receives the value of the maximal reservation from the last round trip.

(c) The token cannot be empty for more than 2 round trips if there are waiting PDUs in the ring.

Note: Recall that in the IEEE 802.5 scheme, the token may be empty for almost 8 round trips while PDUs are waiting for transmission. Moreover, if the number of priority levels is increased, say to 16, the worst case waist becomes 16 round trips in IEEE-802.5, while in our protocol it remains 2.

Proof:

(a) Station \( i \) gets access rights in \( <L_2> \) only if the token is empty. Hence \( R_1 \) contains the maximal \( P_m \) from the previous round trip (Lemma 6). Therefore, we need to prove that \( P_m \geq R_1 \). We have \( \forall m \geq P^* = P \geq R_1 \) where the first inequality and the equality hold because \( <L_2> \) is performed, and the second inequality follows from Lemma 3 since \( T^* = \text{empty} \).
(b) If some station transmits $T^\ast=\text{empty}$, then from Lemma 7 follows that at the next time, this station transmits $P^\ast>R_1^\ast$. From Lemma 3 follows that if $P^\ast>R_1^\ast$, then $T^\ast=\text{full}$, so the claim holds. On the other hand, if $P^\ast=R_1^\ast$ and $T^\ast=\text{empty}$, then from Lemma 6(b) follows that $R_1$ holds the maximal reservation from the last round trip.

(c) Suppose that the token is empty and there are waiting PDUs. From (b) above and Lemma 6(b) follows that after one round trip either $T=\text{full}$ or, if $T=\text{empty}$, then $P=R_1$, where $R_1$ holds the maximal reservation from the last round trip. Since the priority of an empty token cannot increase (Lemma 4(b)), when the station that had set $R_1$ next transmits the frame, holds $T=\text{full}$, whether it receives $T=\text{empty}$ or $T=\text{full}$.

In order to state the fairness property exactly, we need the following:

**Definition**- We say that a station $i$ gets a $p$-opportunity if it gets the opportunity to transmit a PDU with priority $p$, namely if it receives a frame with $T=\text{empty}$ and $P\geq p$ and if it transmits $P^\ast=p^\ast$. Note that when a station gets a $p$-opportunity it does not necessarily transmit a PDU with priority $p$, even if it has such a waiting PDU with this priority. This is because if it has in addition a waiting PDU with a higher priority, it will send the latter.

**THEOREM 3 (fairness)**

Suppose that a station $i$ transmits a PDU with priority $p$ at a time when station $k$ has a waiting PDU with the same priority. Then $k$ will get a $p$-opportunity before $i$ next gets a $p$-opportunity.

**Proof:**

We denote by $t$ the time when $i$ transmits the frame with priority $p$, by $t'$ the time when it receives this frame back after a round trip and by $t''$ the time when $k$ gets a $p$-opportunity for the first time after $t$.

We need to show that after time $t$ and until $t''$, station $i$ does not get a $p$-opportunity.

**Claim (i):** During $[t', t'']$ all stations transmit $P^\ast\geq p$.

**Proof:** First note that at $t'$, when $i$ receives the frame sent at time $t$ back, holds $R_2^\ast\geq p$. This follows from Lemma 6(a), since $Pm_2\geq p$. Therefore, $i$ transmits $P^\ast\geq p$. From that time on, $P$ can be reduced by some station $l$ only in $<L_2>$ or $<A_1>$. We show that in both situations, as long as this happens before $t''$, the
token priority $P$ cannot be reduced to a value below $p$. If in $<L_2>$, holds

$$P^* = V_1[R^+_1 \geq R^+_2 \geq Fm_t \geq p]$$

where the first inequality follows from Lemma 2(a), the second from $<L_1>$, the third from the fact that the token is empty (Lemma 4(a)) and hence $R_1$ holds the maximal reservation (Lemma 6(b)) and the last from the fact that we assume that this happens before $t''$ and hence $k$ has a waiting PDU with priority $p$. The above implies $P^* \geq p$. Similarly, if $P$ is reduced in $<A_1>$, then

$$P^* = V_1[R^+_1 \geq R^+_2 \geq Fm_t \geq p]$$

where the second inequality holds from $<A_2>$ and $<A_1>$ and the third from the fact that $R_1$ contains the maximal reservation (Lemma 6(a)). This completes the proof of Claim (i).

**Corollary 1**: A station $j$ gets a $p$-opportunity during $(t', t'')$ if and only if it receives a frame with $T^\text{empty}$ and $P^- = p$.

**Proof**: Suppose that $j$ gets a $p$-opportunity. By definition holds $T^\text{empty}, P^- = p$ and since the station before $j$ had transmitted the frame at or after $t'$, holds $P^+ \geq p$. Therefore, $P^- = p$. On the other hand, suppose that the station receives $T^\text{empty}$ and $P^- = p$. From Lemma 1, the station is in in LISTEN mode and hence it cannot increase the priority (Lemma 4). Claim (i) shows that it cannot decrease it either, hence $P^* = P^- = p$.

Now we prove that during $(t, t'')$, the stations between $i$ and $k$ get a $p$-opportunity in sequence. This implies that after $i$ gets a $p$-opportunity at time $t$ (and in fact transmits a frame with priority $p$), it does not get a $p$-opportunity before $t''$. To prove the above, we show that if during $(t, t'')$ some station $j \neq k$ gets a $p$-opportunity, its downstream neighbor $j'$ is the next station in the ring to get a $p$-opportunity. Consider the following two cases:

(a) when $j$ gets the $p$-opportunity, it does not send a PDU, i.e. sends $T^* = \text{empty}$. Note that only $i$ gets a $p$-opportunity during $(t, t')$, at time $t$, and at that time $i$ does send a PDU. Therefore, $j$ can get a $p$-opportunity and not send a PDU only during $(t', t'')$. By Corollary 1, it receives $P^- = p$ and by the definition of $p$-opportunity holds $P^* = P^-$. Thus $j$ sends $P^* = p$ and $T^* = \text{empty}$ and its next station downstream $j'$ receives $P^* = p, T^* = \text{empty}$. Corollary 1 shows that $j'$ indeed gets a $p$-opportunity.

(b) when $j$ gets the $p$-opportunity, it does send a PDU, i.e. $T^* = \text{full}$. Note that if $j = i$, it sends the frame with $P^* = P^- = p$, while if $j \neq i$ station $j$ receives the frame with $P^- = p$ (Corollary 1) and thus sends $P^* = p$. In
both cases, the priority of the frame does not change during the round trip (Lemma 4(a)), so \( j \) receives the frame back with \( P^\leq p, T=\text{full} \), but since by now it is time \( t' \) or later, it transmits \( P^\geq p \) (Claim (i)) and \( T^\leq \text{empty} \) (step \(<A_2>\)). If \( j \) transmits \( P^\leq p \), its downstream neighbor receives the frame with \( P^\leq p, T^\leq \text{empty}\) and gets a \( p \)-opportunity (Corollary 1). On the other hand, if \( j \) transmits \( P^\geq p \), we show below that \( j \) is the first station that will later transmit \( P^\leq p \) (no station can transmit \( P^\leq p \) by Claim (i)). At that time, since \( j \) alters \( P \), it cannot send a PDU, thus \( T^\leq \text{empty} \). Its downstream neighbor \( j' \) receives \( P^\leq p, T^\leq \text{empty} \) and thus gets a \( p \)-opportunity (Corollary 1).

To prove that after increasing \( P \) from \( P^\leq p \) to \( P^\geq p \), station \( j \) is the first to transmit \( P^\leq p \), note that when it increases \( P \), station \( j \) sets \( V_j[p] \leftarrow p \) (step \(<A_3>\)). The entry \( V_j[p] \) cannot be altered unless \( j \) transmits a frame with priority \( P^\leq p \), which cannot happen from Claim (i), or with \( P^\leq p \). This is because as long as \( j \) transmits \( P^\geq p \), the entry \( V_j[p] \) cannot change in \( <O_1> \) since \( p \not\in [P^\leq p, P^\geq p) \), or in \( <O_2> \) since \( V_j[p] \leftarrow p \), or in \( <A_4> \) since \( P^\leq p \) (Claim (i)), or in \( <A_5> \) since in this range \( V_j[r] \) becomes or stays \( r \).

From Lemma 5 follows that as long as \( j \) transmits the frame with \( P^\geq p \) and \( V_j[p] \leftarrow p \) holds, the frame cannot be received by any other station whose local vector \( V \) fulfills \( V[p] \leftarrow p \). Now, Lemma 2 implies that \( p \) does not appear in any vector \( V[r] \) \( r \in [0, 7] \) of the local vector at any other station, when the latter receives the frame. Consequently, \( j \) is the only station that can decrease the priority to \( p \) (by executing step \(<L_2> \) or \(<A_7>\)). \( \square \)
References


