Figure 5: Request $R_{n_i}$ is cancelled after the generation of $C_{k_i+1}$

Figure 6: Request $R_{n_i}$ is cancelled before the generation of $C_{k_i+1}$
This cancellation is deleted if an empty slot with $S=1$ arrives on Bus A.

A pair of requests $E_k$ and requests $t_k$ are transmitted.

Cancellation $C_k$ is generated due to the satisfaction of $E_k$ by slot reuse. Later, $C_k$ cancels $R_{n_1}$.

Figure 3: A station with \text{\textit{T-Queue}}, \text{\textit{R-Queue}} and \text{\textit{C-Queue}}

Figure 4: Cancellation $C_{k_1}$ is generated due to the satisfaction of $E_{k_1}$ by slot reuse. Later $C_{k_1}$ cancels $R_{n_1}$.
Figure 1: The Dual Bus configuration

Figure 2: A system to demonstrate the motivation for the improved scheme
arrives at any \( T\text{-Queue}_l, \ 1 \leq l \leq J-1 \). By Claim A.3, station \( j \) releases an empty slot for \( R_i \) on Bus \( A \). Let this slot be \( S_{R_i}^j \). By Lemma A.1 \( S_{R_i}^j \) arrives at station 2 when \( R_i \) is at the head of \( T\text{-Queue}_2 \) or \( R_i \) is already not in \( T\text{-Queue}_2 \). In either case it is clear that station 2 releases an empty slot for \( R_i \). Using Lemma A.1 now for stations 3, 4, ..., \( J-1 \), it is clear that station \( J-1 \) releases an empty slot for \( R_i \) which arrives at \( J \) when \( E_i \) is at the head of \( T\text{-Queue}_J \) or it is not there already. Therefore it is clear that \( J \) receives an empty slot for the transmission of its packet.
Because the request is then removed from the queue, and because the queue uses a FIFO discipline, the claim follows.

In the following lemma the ordinal numbers of the entries and requests are in relation to station I+1.

**Lemma A.1:** Let $S_{R_j}^I$ be an empty slot on Bus A that a station I released for request $R_j$ at the head of its T-Queue. Then, at the time when $S_{R_j}^I$ arrives at I+1 either entry $E_j$ is already not in T-Queue$_{I+1}$ or it is at the head of T-Queue$_{I+1}$.

**Proof:** The proof is by induction on the entries in T-Queue$_{I+1}$. We assume that the network begins its operation when all the queues are empty. Let $E_1$ be the first entry in T-Queue$_{I+1}$. By Claim A.1, $R_1$ eventually appears in T-Queue$_J$. If station I releases an empty slot for $E_1$ then this slot arrives at station I+1 empty and either $E_1$ is still at the head of T-Queue$_{I+1}$ or it was deleted before.

Assume correctness for the first $n$ entries in T-Queue$_{I+1}$. We now prove correctness for the $n + 1^{th}$ entry. By Claims A.1 and A.2 the first $n + 1^{th}$ entries in T-Queue$_{I+1}$ also appear eventually in T-Queue$_J$ and in the same relative order. Therefore, if station I releases an empty slot for the $n + 1^{th}$ entry, it does so only after releasing an empty slot for each of the first $n$ entries in T-Queue$_{I+1}$. By the induction assumption concerning the $n^{th}$ entry, if I releases an empty slot for this entry, this slot leaves station I+1 when the $n^{th}$ entry is not in T-Queue$_{I+1}$ anymore. Thus, if I releases an empty slot for the $n + 1^{th}$ entry also, this slot arrives at I+1 when the $n + 1^{th}$ entry is at the head of T-Queue$_{I+1}$ or it is not in T-Queue$_{I+1}$ anymore.

**Proof of Theorem A.1:** First notice that any packet of station I is transmitted since station I receives an infinite stream of empty slots and so it eventually serves any entry in its T-Queue.

Consider now a packet of a station $J$, $J > I$, that is submitted for transmission. Assume that the packet is represented by the entry $E_i$ in T-Queue$_J$. By Claim A.1 $R_i$ eventually
Appendix A

In this Appendix we specify the proof for lack of starvation in the important case of DQDB without slot reuse. It is derived from the proofs in Section 5. We prove the following Theorem:

**Theorem A.1:** DQDB ensures lack of starvation (i.e. any packet of a station \(J\) is guaranteed to be transmitted).

In order to prove Theorem A.1 we define and prove several claims.

**Claim A.1:** If entry \(E_j\) appears in \(T\_Queue_I\), then \(R_j\) will appear in the T-Queue of any of the stations \(1..I-1\).

**Proof:** Entry \(E_j\) is either a request or a packet. In either case, at the time when it entered \(T\_Queue_I\), \(R_j\) entered \(R\_Queue_I\). Therefore, by the description in Section 4, \(R_j\) will eventually be transmitted upstream, will arrive at station \(I-1\), will enter \(T\_Queue_{I-1}\) and \(R\_Queue_{I-1}\) and so on, until station \(I\).

**Claim A.2:** If entry \(E_i\) appears in \(T\_Queue_I\) before entry \(E_j\), then \(R_i\) appears before \(R_j\) in the T-Queue of any of the stations \(1..I-1\).

**Proof:** Since \(E_i\) appears in \(T\_Queue_I\) before \(E_j\), then \(R_i\) entered \(R\_Queue_I\) before \(R_j\). Because of the FIFO in the queues and on the bus, \(R_i\) is transmitted by \(I\) before \(R_j\) and so \(R_i\) arrives at \(I-1\) before \(R_j\). By continuing this claim until station \(I\), the claim follows.

**Claim A.3:** Station \(I\) leaves an empty slot to pass on Bus A for every request in its T-Queue.

**Proof:** Station \(I\) receives an infinite stream of empty slots. For every slot it checks the entry at the head of its T-Queue and does not transmit into the slot if this entry is a request.
that the total throughput gain in the various schemes are 1.20, 1.28, 1.30 and 1.40 for schemes 1, 2, 3 and 4 respectively. We see that the improved scheme of Section 3 achieves a throughput gain higher than the scheme of [5] by about 17%. In higher inter-arrival rates the differences among the schemes are less significant.

References


deleted from $T_{\text{Queue}_{i+1}}$, these slots ensure the satisfaction of $E_j$.

We can now complete the section with the proof of Theorem 1:

**Proof of Theorem 1:** By Claim 3 station $i$ satisfies every request that arrives from station 2. By Lemma 4 this means that every entry in $T_{\text{Queue}_2}$ is satisfied, including all the requests that arrive from station 3. Therefore, by applying Lemma 4 inductively, every entry in every station is satisfied, and thus every packet is eventually transmitted.

Before concluding this section we mention that the above also proves that the scheme in case of Erasure Nodes also ensures lack of starvation. The difference between the scheme for Destination Release and the one for Erasure Nodes is that in the case of Erasure Nodes a slot with a set S bit can delete cancellations that otherwise, if the S bit could have been reset by every station, would have not been deleted. Also, a non erasure station with $ES > 0$ can only delete self requests. It can be easily verified that claims 1-4 and lemmas 1-4 hold for this case also.

6 Simulation results

We have computed by simulation several versions of the scheme in [5] and the improved scheme of Section 3. We have simulated the scheme of [5], the scheme of [5] with the improvement of the S bit only, the scheme of [5] with the improvement that a destination station does not perform slot release if the released slot is not used to satisfy a request or for a self transmission and the last simulated scheme is the one of Section 3. The simulated schemes will be denoted by schemes 1, 2, 3 and 4 respectively.

We have simulated a system with 50 stations evenly located along the bus with three slots between adjacent stations. We assume that stations generate messages according to a Poisson process with identical rates, all messages are one slot long and the source-destination traffic is uniform.

When simulating the system with a mean message inter-arrival time of 0.125, we receive
time also and \( R_{n_1}, \ldots, R_{n_i} \) are included in \( R_1, R_2, \ldots, R_{k_{i+1}} \). Clearly, only empty slots that satisfy requests \( R_1, \ldots, R_{k_{i+1}} \) can arrive at \( I + 1 \) until \( t_{k_{i+1}} \). This is also true for any empty slot \( S_j \) by which \( I \) satisfies a request \( R_j \) and that arrives during \( [t_{k_{i+1}}, t_{n_{i+1}}] \) because during this interval \( ES_{I+1} > REQ_{I+1} \) and so \( S_j \) deletes a cancellation with an index smaller than \( k_{i+1} \). By Lemma 2 \( j < k_{i+1} \) also. Therefore, at most \( k_{i+1} - (i + 1) \) empty slots by which \( I \) satisfies a request with an index smaller than \( n_{i+1} \) can arrive until \( t_{n_{i+1}} \). Since \( (n_{i+1} - (i + 1)) \) requests with such an index are sent to \( I \), the lemma follows.

If entry \( E_{k_{i+1}} \) is in \( T_{\text{Queue}}_{I+1} \) at \( t_{n_i} \) then assume that at time \( t_{n_i} \) entries \( E_i, \ldots, E_{k_{i+1}} \) are yet in \( T_{\text{Queue}}_{I+1} \), where \( l \geq k_{i+1} \). Assume that during \( [t_{n_i}, t_{k_{i+1}}] \) \( x \) entries in \( T_{\text{Queue}}_{I+1} \) are satisfied by arriving empty slots. Clearly, \( x \leq k_{i+1} - l \) since \( E_{k_{i+1}} \) is still in \( T_{\text{Queue}}_{I+1} \) at \( t_{k_{i+1}} \). It turns out that out of entries \( E_{k_{i+1}}, \ldots, E_{k_{i+1}} \) at most \( (k_{i+1} - k_i - x) \) could have been satisfied by slot reuse and could cause the generation of cancellations. Since during \( [t_{k_{i+1}}, t_{n_{i+1}}] \) holds \( ES_{I+1} > REQ_{I+1} \) and \( C_k \) is not in \( C_{\text{Queue}}_{I+1} \), it turns out that during \( [t_{k_{i+1}}, t_{n_{i+1}}] \) at most \( (k_{i+1} - k_i - x - 1) \) slots by which \( I \) satisfies requests can arrive at \( I + 1 \). Therefore, during \( [t_{n_i}, t_{n_{i+1}}] \) at most \( (k_{i+1} - k_i - x - 1) + x = (k_{i+1} - k_i - 1) \) such slots can arrive at \( I + 1 \), each of them satisfying a request with an index smaller than \( n_{i+1} \). By the induction assumption, after \( t_{n_i} \) at least \( n_i - k_i \) empty slots by which \( I \) satisfies requests with an index smaller than \( n_i \) will arrive at \( I + 1 \) and to this number we add the additional \( (n_{i+1} - n_i - 1) \) requests that are sent to \( I \) and are for entries with an index higher than \( n_i \) but smaller than \( n_{i+1} \). Therefore, after \( t_{n_{i+1}} \) at least \( n_i - k_i + (n_{i+1} - n_i - 1) - (k_{i+1} - k_i - 1) = n_{i+1} - k_{i+1} \) empty slots by which \( I \) satisfies requests with an index smaller than \( n_{i+1} \) will arrive at \( I + 1 \).

\[\square\]

**Lemma 4:** If station \( I \) releases an empty slot for every request in \( T_{\text{Queue}}_I \) then every entry in \( T_{\text{Queue}}_{I+1} \) is satisfied.

**Proof:** Consider entry \( E_j \), \( j \geq 1 \). By Claim 1 request \( R_j \) is either cancelled at \( I + 1 \) or it arrives at \( T_{\text{Queue}}_I \). If \( R_j \) arrives at \( T_{\text{Queue}}_I \) then it is satisfied and by Lemma 1 entry \( E_j \) is also satisfied. If \( R_j \) is canceled at station \( I + 1 \) assume that it is canceled at time \( t \) by a cancellation \( C \) that was generated due to the satisfaction by slot reuse of entry \( E_k \). By Lemma 3 at least \( j - k \) empty slots will arrive at \( I + 1 \) after time \( t \) and since at \( t \) entry \( E_k \) is
Lemma 3: Assume that at time $t_{n_i}$ request $R_{n_i}$ is the $i^{th}$ request in $R_{\text{Queue}_{I+1}}$ that is canceled by a cancellation $C_{k_i}$ and that $C_{k_i}$ was generated due to the satisfaction by slot reuse of entry $E_{k_i}$. Then, after $t_{n_i}$ at least $n_i - k_i \geq 0$ (by Claim 4) empty slots by which station $I$ satisfies requests with an index smaller than $n_i$ will arrive at $I + 1$.

Proof: By induction on $i$. Consider $R_{n_i}$. Assume that at time $t_{k_i}$ cancellation $C_{k_i}$ is generated and that $C_{k_i}$ was generated due to the satisfaction by slot reuse of entry $E_{k_i}$. Then, after $t_{n_i}$ at least $n_i - k_i$ empty slots by which station $I$ satisfies requests with an index smaller than $n_i$ will arrive at $I + 1$.

By the FIFO in $R_{\text{Queue}_{I+1}} n_i - 1$ requests have been sent to $I$ until $t_{n_i}$. Until $t_{k_i}$ at most $k_i - 1$ empty slots by which $I$ satisfies requests could arrive at $I + 1$ and therefore, if during $[t_{k_i}, t_{n_i}]$ no empty slot by which $I$ satisfies a request is passing at $I + 1$, then after $t_{n_i}$ at least $n_i - k_i$ such empty slots will arrive at $I + 1$. If during $[t_{k_i}, t_{n_i}]$ at least one such empty slot is passing, consider the last one. Denote this slot by $S_i$ and assume that it satisfies request $R_i$ at $I$. Since during $[t_{k_i}, t_{n_i}]$ holds $EC_{I+1} > REQ_{I+1}$, $S_i$ deletes a cancellation $C'$ in $C_{\text{Queue}_{I+1}}$ such that $C'$ was generated due the satisfaction by slot reuse of an entry $E_p$, $p < k_i$. Therefore, by Lemma 2, $l \leq p < k_i$. Thus, in this case also at most $k_i - 1$ empty slots by which station $I$ satisfies requests can arrive at $I + 1$ until $t_{n_i}$. Finally, it is clear that all the considered empty slots that arrive after $t_{n_i}$ satisfy a request with an index smaller than $n_i$.

Assume correctness for requests $R_{n_1}, ..., R_{n_i}$. Consider $R_{n_{i+1}}$, $i + 1 > 1$. We divide the discussion into two cases: $t_{n_i} > t_{k_{i+1}}$ and $t_{n_i} < t_{k_{i+1}}$. The first case is depicted in Figure 5. By the induction assumption, in this case at least $n_i - k_i$ empty slots by which $I$ satisfies requests with indexes smaller than $n_i$ will arrive at $I + 1$ after $t_{n_i}$. Notice that in this case $n_{i+1} = n_i + 1$ and therefore at least $n_{i+1} - k_i - 1$ empty slots by which $I$ satisfies requests with an index smaller than $n_{i+1}$ will arrive at $I + 1$ after $t_{n_i}$. During $(t_{n_i}, t_{n_{i+1}}]$ at most $k_{i+1} - k_i - 1$ such empty slots can arrive at $I + 1$ because during this interval $EC_{I+1} > REQ_{I+1}$, $C_{k_{i+1}}$ is in $C_{\text{Queue}_{I+1}}$ but $C_k$ already not and when $EC > REQ$ every arriving empty slot that satisfies a request deletes a cancellation from $C_{\text{Queue}}$. The lemma follows.

In the second case we assume that $t_{n_i}$ is before $t_{k_{i+1}}$, as depicted in Figure 6.

If entry $E_{k_{i+1}}$ is not in $T_{\text{Queue}_{I+1}}$ at $t_{n_i}$ yet, than $R_{k_{i+1}}$ is still not in $R_{\text{Queue}_{I+1}}$ at this
for $S_{n_1}$, at most $(p-(i-1))$ cancellations that later cancel requests could have been generated out of entries $E_1,...,E_p$. In this case, and taking into account that $S_{n_1}$ deletes a cancellation and/or satisfies an entry in $T_{\text{Queue}_{I+1}}$ which is therefore not being satisfied by slot reuse and does not cause the generation of a cancellation, the total number of generated cancellations out of $E_1,...,E_i$ that could have been used later to cancel requests is $(p-(i-1))+(l-p)-1 = l-i < n_{i+1}-(i+1)$, a contradiction. Therefore, assume that $S_{n_{i-1}}$ either deletes a cancellation in $C_{\text{Queue}_{I+1}}$ and/or it satisfies an entry in $T_{\text{Queue}_{I+1}}$. 

Continuing with $S_{n_{i-2}},...,S_{n_1}$ by the same way, we receive that out of entries $E_1,...,E_i$ at most $(l-i) < n_{i+1}-(i+1)$ cancellations could have been used to cancel requests, a contradiction. Therefore, when $S_{n_{i+1}}$ arrives at $T_{\text{Queue}_{I+1}}$, entry $E_{n_{i+1}}$ is not in the queue or it is in its head.

Lemma 2: Assume that station $I$ satisfies every request in $T_{\text{Queue}_{I}}$ that arrives from station $I+1$. Assume that $R_{n_i}$ is the $i^{th}$ request that station $I$ satisfies, by the empty slot $S_{n_i}$. Assume also that $S_{n_i}$ deletes cancellation $C_{k_i}$ in $C_{\text{Queue}_{I+1}}$ and that $C_{k_i}$ has been generated due to the satisfaction by slot reuse of entry $E_{k_i}$. Than $k_i \geq n_i$.

Proof: First consider $S_{n_1}$. If $n_1 = 1$ and $S_{n_1}$ deletes a cancellation then the lemma is clear. If $n_1 > 1$ then by the FIFO of $R_{\text{Queue}_{I+1}}$ requests $R_1,...,R_{n_i-1}$ have been cancelled at $I+1$ before the arrival of $S_{n_1}$. If $S_{n_1}$ deletes cancellation $C_{k_1}$ such that $k_1 \leq n_1 - 1$, this means that at most $k_1 - 1 < n_1 - 1$ cancellations could have been used to cancel requests until the arrival of $S_{n_1}$ at $I+1$, a contradiction. Therefore, $k_1 \geq n_1$.

Next consider $S_{n_{i+1}}, i+1 > 1$. If $S_{n_{i+1}}$ does not delete a cancellation we are done. Therefore assume that $S_{n_{i+1}}$ deletes cancellation $C_{k_{i+1}}$ in $C_{\text{Queue}_{I+1}}$ and assume by contradiction that $k_{i+1} \leq n_{i+1} - 1$.

By the FIFO in $R_{\text{Queue}_{I+1}}$ it is clear that until the time when $S_{n_{i+1}}$ arrives at $I+1$, at least $n_{i+1} - (i+1)$ entries out of $E_1,...,E_{k_{i+1}-1}$ have been satisfied by slot reuse and the corresponding generated cancellations have been used to cancel requests. We take $E_{k_{i+1}-1}$ because $C_{k_{i+1}}$ is cancelled by $S_{n_{i+1}}$. However, following the same proof as that of Lemma 1, since $k_{i+1} - 1 < n_{i+1}$ this is impossible, a contradiction.
is already not in it.

**Proof:** By induction on \( i \). Consider \( S_{n_i} \). By the FIFO of \( R_{\text{Queue}_{i+1}} \) requests \( R_1, R_2, \ldots, R_{n_i} \) are cancelled at \( I + 1 \) and do not arrive at \( I \). Therefore, at least \( n_i - 1 \) cancellations of requests have been occurred at \( I + 1 \) before the arrival of \( S_{n_i} \). Since a cancellation is generated due to the satisfaction of an entry in the \( T_{\text{Queue}} \) by slot reuse, it follows that at least \( n_i - 1 \) entries in \( T_{\text{Queue}_{i+1}} \) have been satisfied by slot reuse and therefore entries \( E_1, E_2, \ldots, E_{n_i - 1} \) are already not in \( T_{\text{Queue}_{i+1}} \) when \( S_{n_i} \) arrives at \( I + 1 \).

Assume correctness for \( R_{n_1}, \ldots, R_{n_i} \). Consider \( R_{n_i+1} \), \( i + 1 > 1 \). Assume by contradiction that when \( S_{n_i+1} \) arrives at \( I + 1 \), entry \( E_{i+1} \) is at the head of \( T_{\text{Queue}_{i+1}} \), \( l + 1 < n_{i+1} \). It is clear that out of entries \( E_1, \ldots, E_i \) at least \( n_{i+1} - (i + 1) \) entries have been satisfied by slot reuse and the corresponding generated cancellations have been used to cancel \( n_{i+1} - (i + 1) \) requests in \( R_{\text{Queue}_{i+1}} \). This is because the FIFO in \( R_{\text{Queue}_{i+1}} \) implies that \( R_{n_{i+1}} \) is sent to \( I \) only after \( R_{n_{i+1}-1}, \ldots, R_n \) were sent to \( I \) or were cancelled in \( I + 1 \). Consider \( S_{n_i} \) and assume that when it arrives at \( I + 1 \), \( T_{\text{Queue}_{i+1}} \) is empty and that \( S_{n_i} \) does not delete a cancellation in \( C_{\text{Queue}_{i+1}} \), i.e. \( EC_{i+1} \leq \text{REQ}_{i+1} \) at the arrival time of \( S_{n_i} \) at \( I + 1 \). Let \( E_s \) be the last entry in \( T_{\text{Queue}_{i+1}} \) before the arrival of \( S_{n_i} \). By the induction assumption \( s \geq n_i \).

Since \( S_{n_i} \) does not delete a cancellation from \( C_{\text{Queue}_{i+1}} \) and \( T_{\text{Queue}_{i+1}} \) is empty when it arrives at \( I + 1 \), all the cancellations that were generated due to the satisfaction of entries in \( E_1, \ldots, E_s \) by slot reuse have been already used to cancel requests or have been deleted. By the FIFO in \( R_{\text{Queue}_{i+1}} \) and since \( s \geq n_i \), at least \( i \) requests out of \( R_1, \ldots, R_s \) have not been cancelled by cancellations. Therefore, at most \( (s - i) \) cancellations that later cancel requests could have been generated out of entries \( E_1, \ldots, E_s \). Assume that all the entries \( E_{s+1}, \ldots, E_i \) are satisfied by slot reuse and all the corresponding cancellations are used to cancel requests. Thus, the total possible number of generated cancellations out of \( E_1, \ldots, E_i \) that could have been used to cancel requests is \( (s - i) + (l - s) = l - i < n_{i+1} - 1 - i = n_{i+1} - (i + 1) \), a contradiction. Therefore, assume that \( S_{n_i} \) either deletes a cancellation in \( C_{\text{Queue}_{i+1}} \) and/or it finds \( T_{\text{Queue}_{i+1}} \) not empty and satisfies an entry.

Next consider \( S_{n_{i-1}} \). Assume that \( S_{n_{i-1}} \) does not delete a cancellation in \( C_{\text{Queue}_{i+1}} \) and that \( T_{\text{Queue}_{i+1}} \) is empty when it arrives at \( I + 1 \). Let \( E_p \) be the last entry in \( T_{\text{Queue}_{i+1}} \) before the arrival of \( S_{n_{i-1}} \). By the induction assumption \( p \geq n_{i-1} \). Using similar arguments as
at the head of its T.Queue and does not transmit into the slot if this entry is a request. Because the request is then removed from the queue and because of the FIFO in the queue, the claim follows.

Claim 3: Assume that entry E_i in station I is satisfied at time t and let s be the transmission time of a slot. Then at time t + s request R_i is not in R.Queue_I.

Proof: By induction on i. Concerning E_1 assume that it is satisfied at time t. Clearly, R_1 entered R.Queue_I before t. During the interval [t,t + s] an R bit is passing on Bus B by which R_1 can be transmitted or deleted from R.Queue_I as a part of a pair.

Assume correctness for E_1,...,E_i. We now show correctness for E_{i+1}. Assume that E_{i+1} is satisfied at time t. Clearly, R_{i+1} entered R.Queue_I before t. The latest time before t when E_i could have been satisfied is t - s. Thus, by the induction assumption R_1,...,R_i are already not in R.Queue_I at t. Therefore, R_{i+1} can be transmitted or deleted from R.Queue_I as a part of a pair by the R bit that passes on Bus B during [t,t + s].

Claim 4: Assume that cancellation C in station I is created due to the satisfaction of entry E_k by slot reuse. Also assume that later C cancels request R_j. Then j >= k.

Proof: Assume that C is created and thus that E_k is satisfied at time t. By Claim 3 at time t requests R_1,...,R_{k-1} are already not in R.Queue_I and the claim follows.

In lemmas 1-4 we refer to two stations, I and I+1. However, we only consider the entries in T.Queue_{I+1}, their corresponding requests in R.Queue_{I+1} and the cancellations in C.Queue_{I+1}. Therefore, in these lemmas all the ordinal numbers of the entries and requests are those given in I+1. Notice also that all the requests in T.Queue_I arrived at T.Queue_I from R.Queue_{I+1}

Lemma 1: Assume that station I satisfies every request in T.Queue_I that arrives from station I+1. Assume that R_{n_i} is the i'th request that I satisfies, by the empty slot S_{n_i}. Then, when S_{n_i} arrives at station I+1, entry E_{n_i} of T.Queue_{I+1} is either at the head of the queue or it
the request related to $E_2$ and so on. Notice that the description above does not change the scheme of Section 3. It only gives an interpretation to the operations described there.

A station with its three queues is depicted in Figure 3. An $E$ in the $T$ Queue stands for a request or a packet for transmission, an $R$ in $R$ Queue stands for a request and a $C$ in the $C$ Queue stands for a cancellation which was generated due to the satisfaction by slot reuse of an entry in $T$ Queue.

Finally, the scheme for the case of Erasure Nodes can be easily described as above.

5 Correctness proof

In this section we prove for the case of Destination Release that the improved scheme of Section 3 ensures lack of starvation. This claim is summarized in Theorem 1:

**Theorem 1:** The improved 'cancelled requests' scheme for slot reuse ensures lack of starvation.

First notice that in the improved scheme we enable the case that a station does not release a slot destined to itself. This slot can be later released and reused by any downstream station. Therefore, in this case we can assume that the slot carried information to the releasing station and thus this improvement does not have any influence on the correctness proofs. In order to prove the Theorem we first prove four claims and four lemmas.

**Claim 1:** Every request $R_j$ in $R_{Queue_{I+1}}$ is either canceled by a cancellation in $I + 1$ or it eventually appears in $T_{Queue_I}$.

**Proof:** By the description in Section 4, $R_j$ eventually arrives at the head of $R_{Queue_{I+1}}$ and it is either canceled by a cancellation if it is a part of a pair or it is transmitted to $I$ and enter $T_{Queue_I}$.

**Claim 2:** Station $I$ leaves an empty slot to pass on Bus A for every request in $T_{Queue_I}$.

**Proof:** Station $I$ receives an infinite stream of empty slots. For every slot it checks the entry
If a set $R$ bit arrives on Bus $B$, a request is inserted into $T_{Queue}$ and $R_{Queue}$ and the bit is reset. Then, and also in the case that the $R$ bit arrives unset, if both $C_{Queue}$ and $R_{Queue}$ are not empty, the first cancellation and request in these queues are deleted respectively and the bit continues unset. If $C_{Queue}$ is empty but $R_{Queue}$ is not, the $R$ bit is set and the first request in $R_{Queue}$ is deleted.

To specify which cancellation is deleted when an empty slot with a set $S$ bit is arriving on Bus $A$, we divide the cancellations and requests in $C_{Queue}$ and $R_{Queue}$ respectively into \textit{pairs}. If $C_{Queue}$ and $R_{Queue}$ are not empty, the first request and first cancellation in these queues are said to be a pair. The second request and cancellation are also a pair and so on. When a cancellation is inserted into $C_{Queue}$ and there is a request in $R_{Queue}$ which is not a part of a pair yet, the cancellation and the first such request become a pair. A symmetric operation takes place when a request is inserted into $R_{Queue}$. Notice that a pair is actually a request and a cancellation that cancels it because by the description above every passing slot on Bus $B$ deletes the pair consisting the first request and cancellation in the $R_{Queue}$ and $C_{Queue}$ respectively and as we describe shortly a cancellation which is a part of a pair is never deleted from $C_{Queue}$ by an empty slot arriving on Bus $A$.

Using the above description, an empty slot that deletes a cancellation because there are more cancellations than requests, deletes the first cancellation that is not a part of a pair. Notice that it can happen that the first cancellation in $C_{Queue}$ is not deleted in this case.

In the following we will say that a cancellation cancels a request at the time when they become a pair. We can say that because it is guaranteed that the cancellation will not be deleted by an empty slot arriving on Bus $A$ and eventually the pair will be deleted from the queues.

Finally, in the following we will not distinguish between a request or a packet in $T_{Queue}$ and will denote both by the term \textit{entry}. We number the entries in $T_{Queue}$ according to the order of their entrance into the queue. Thus, $E_1$ denotes the first entry in $T_{Queue}$, $E_2$ denotes the second entry and so on. Notice that by the assumption above that every arriving request in a set $R$ bit on Bus $B$ is inserted both into $T_{Queue}$ and $R_{Queue}$, than for every entry in $T_{Queue}$ there is a corresponding request in $R_{Queue}$. The corresponding requests will receive the same numbers as their entries, thus $R_1$ is the request related to $E_1$, $R_2$ is
if a S bit is set, it remains set at least until it arrives at the next Erasure Node and it can cause the deletion of cancellations that otherwise were not deleted.

Concerning the case where a destination station does not release a full slot if it is not used to satisfy a request or for a self transmission, it is not worth while that the Erasure Node will not release such a slot because in such a case the slot will not be reused at least until it arrives at the next Erasure Node. In order to overcome this difficulty it is possible to use an additional bit in the ACF which an Erasure Node sets instead of not releasing the slot. An empty slot with this bit set and with S=0 signals that it shall be treated as to a full slot that is released by the first station that uses it. Notice that when a non erasure station uses such a slot, it increments its ES. ES in such a station can be used to cancel self requests only.

4 Notation

For the proofs in the next section we represent the counters in a station (in relation to transmissions on one bus) by three queues, T.Queue (Transmission Queue), R.Queue (Request Queue) and C.Queue (Cancellation Queue). The queues in station \( I \) will be denoted by a subscript \( I \).

The counters CD and RQ are represented by the T.Queue, REQ is represented by the R.Queue and ES is represented by C.Queue. We can summarize our improved scheme for the case of Destination Release using the presentation by the queues as follows: When a station submits a packet for transmission it inserts a packet into T.Queue and a request into R.Queue. Every empty slot arriving on Bus A satisfies the first entry in T.Queue. If it is a packet, the packet is transmitted. If it is a request, the slot continues empty and the request is deleted from T.Queue. The S bit in the slot is set.

Every arriving full slot on Bus A that contains data destined to the station is released if T.Queue is not empty. In this case the slot is used to satisfy the first entry in T.Queue and a cancellation is inserted into C.Queue. If the slot satisfies a request, its S bit is set. Also, when an empty slot arrives on Bus A with a set S bit and there are more cancellations in C.Queue then requests in R.Queue, a cancellation is deleted from the C.Queue. We describe later which cancellation is deleted.
ting on Bus A and our scenario begins when all the slots on Bus A are full with information from station $I$ to station $M$. Consider now the following scenario:

1. Station $I$ receives $x$ requests on Bus B.

2. No more requests arrive on Bus B and $i$ performs $x$ slots release. We arrive to the situation where $ES = x$ and $REQ = 0$ at $I$.

3. The following begins to happen continuously: $I$ receives requests in every slot on Bus B. $I$ performs slot release in every slot passing on Bus A. $I$ schedules self packets for transmission. Assume that the first request on Bus B arrives before the first slot release so that every slot release increments ES. By lines A34-A40 in the algorithm, $I$ does not transmit its own requests and so its REQ increases with every self packet scheduled for transmission. Also, every request arriving on Bus B is cancelled, causing ES to be decremented. However, ES is incremented immediately again with the slot reuse on Bus A. Thus ES has only two possible values, $x$ or $x - 1$.

Assume that the above terminates when $REQ = y$, $y < x$, i.e. no more requests arrive on Bus B, no more slot release is performed on Bus A and $I$ does not schedule more packets for transmission.

4. $I$ receives $x$ empty slots with $S=1$ on Bus A. Assume that it does not decrement ES at least when one such a slot is passing.

5. Consider the situation after all the $x$ empty slots that passed at $I$ had left Bus A. The ES at $I$ is still positive. Also assume that Bus A continues to be full with information from station $I$ to $M$. If later a request arrives at $I$ from a downstream station on Bus B, it will be cancelled and the requesting station will be starved.

### 3.3 The new scheme in the case of Erasure Nodes

Concerning the implementation of the scheme in the case of Erasure Nodes, we must notice that non erasure stations are not capable to change the value of a bit from '1' to '0'. Therefore,
for a self transmission or to satisfy a downstream request. The station does not increment ES otherwise because this can clearly lead to starvation. The released slot in this case is not counted and upstream stations are not aware of the slot release. In [11] it was suggested to take this case into account also by not allowing the destination station to release the slot. The slot will be reused/released by a downstream station only if it is used for a self transmission or to satisfy a downstream request.

3.2 The new scheme

As mentioned, the improved scheme needs an additional bit in the ACF. We denote this bit by S for Satisfying. This bit is generated by the slot generator with a '0' value. The S bit has significance only when the slot is empty (B=0). When a station releases an empty slot, it distinguishes between two cases. If the slot is used to satisfy a request, the S bit is set to '1'. Otherwise, it is set to '0'. When a slot is full, the value of S is not important and it is not read.

Also, as before, when a slot is released and it is used for a self transmission or to satisfy a request, ES is incremented. ES is decremented when positive in two cases. First when a request is cancelled and second, when ES>REQ and an empty slot with a set S bit is arriving. Notice that ES is not decremented in the case when an empty slot with an unset S bit arrives.

The pseudo code for this scheme is as the one in Section 2 except that lines A3, A8, A14 and A23 are modified to include the improvements. The new lines appear below:

A3 if ((B^-_{Bus_A} = 0) and (S^-_{Bus_A} = 1) and (ES>REQ)) then decrement ES
A8 \( B^+_{Bus_A} = 0, S^+_{Bus_A} = 1 \)
A14 \( B^+_{Bus_A} = 0, S^+_{Bus_A} = 1 \)
A23 if (B^-_{Bus_A} = 0) then S^+_{Bus_A} = 0

We now show a scenario for starvation that occurs because a station with ES>REQ does not decrement its ES when an empty slot with S=1 arrives. Thus, our scheme is optimal in this sense.

Consider again the system of Figure 2. Assume that station 1 is the only station transmit-
3 The improved 'cancelled requests' scheme for slot reuse

3.1 Motivation

In the following we assume the case of Destination Release. We will later handle the case of slot reuse by Erasure Nodes. We demonstrate the motivation for our modified scheme by the system of Figure 2.

Assume that station $I$ is the only station transmitting on Bus A, it is overloaded and continuously transmitting. Assume that all the slots downstream from station $I$ are full with data to station $M$ when station $L$ wants to transmit a message. $L$ transmits a request on Bus B which causes station $I$ to increment its RQ. Then, assume that station $I$ releases one slot that station $I$ transmitted to it. It decrements RQ and increments ES. Assume now that later station $I$ releases an empty slot to satisfy the request from $L$ and that this slot travels on Bus A until its end without being used. Notice that $L$ used the released slot for its transmission. Assume now that station $K$ requests a slot for transmission. If station $I$ did not decrement its ES due to the empty slot that station $I$ has released to satisfy the request from $L$, the request of $K$ will be cancelled by $I$ and if station $I$ continues to be overloaded, station $K$ will be starved. Therefore, it was suggested that ES will be decremented with every passing empty slot [10].

However, notice that if station $I$ stops transmitting before it releases the empty slot to satisfy the request from $L$, station $I$ will decrement ES before the empty slot by which $I$ satisfies the request from $L$ is arriving. Such a decrement however, is too early since meantime, until the released slot from station $I$ is arriving, the cancellation could have been used to cancel another request arriving from downstream stations, thus improving efficiency.

Therefore, in our new scheme we distinguish between empty slots that satisfy downstream requests and such that do not. It turns out that it is enough that a station will decrement ES only when ES $>$ REQ and when an empty slot that satisfies a request is arriving. The implementation requires an additional bit in the ACF to distinguish between the two types of empty slots.

Also, in the scheme of [5] a station increments ES only when it releases a slot that is used
else if (message=TRUE) then begin
    transmit message
    if (B_{Bus_A}⁻ = 1) then increment ES
    B_{Bus_A}⁺ = 1
    end
else
    B_{Bus_A}⁺ = 0
end

if (R_{Bus_B}⁻ = 0) then (* Operations on Bus B *)
if ((REQ>0) and (ES>0)) then begin
    decrement ES
    decrement REQ
    end
else if (REQ>0) then begin
    decrement REQ
    R_{Bus_B}⁺ = 1
    end
else begin
    increment RQ
end
increment RQ
if (ES>0) then begin
    decrement ES
    R_{Bus_B}⁺ = 0
    end
end
generates a cancellation which is counted in ES. The cancellations in ES are used to cancel requests to upstream stations as follows: If an unset R bit arrives on Bus B than if REQ > 0 and ES > 0, both counters are decremented and the R bit is left unset. If however, ES = 0 but REQ > 0, REQ is decremented and R is set. If the R bit arrives set, than if ES > 0 then it is reset and ES is decremented. Otherwise it is not changed. In both cases RQ is incremented.

The scheme described so far can lead to starvation as we later show. In order to avoid this possibility, ES is decremented with every arriving empty slot on Bus A and when positive [10]. This scheme ensures lack of starvation but this claim has not been proved before.

A pseudo code for the scheme is given below. It describes the transmissions on Bus A. Those on Bus B are symmetric. We denote the value of a bit when arriving and leaving a station with the superscripts ‘+’ and ‘-’ respectively. By a subscript we denote the Bus on which the bit arrives. We also use the boolean variable message which is true when a station has a packet for transmission and it is false otherwise. In the code we do not address the operations being performed when a station schedules a message for transmission. They are similar to those described for DQDB.

Algorithm for station i:

A1 if ((B_{Bus A}^- = 0) or ((B_{Bus A}^+ = 1) and (Destination=i))) (* Operations on Bus A *)
A2 then begin
A3 if ((B_{Bus A}^- = 0) and (ES > 0)) then decrement ES
A4 if ((message=FALSE ) and (RQ > 0))
A5 then begin
A6 decrement RQ
A7 if (B_{Bus A}^+ = 1) then increment ES
A8 B_{Bus A}^+ = 0
A9 end
A10 else if ((message=TRUE) and (CD > 0))
A11 then begin
A12 decrement CD
A13 if (B_{Bus A}^- = 1) then increment ES
A14 B_{Bus A}^- = 0
A15 end
starvation.

A simple way to improve the performance of DQDB is to enable slot reuse. In this method a full slot can be reused again after passing its destination. Two methods were suggested to implement this idea. In the first one, called Destination Release, every station is capable to reuse or release full slots that carry information to itself. In the second one only several specified stations along the bus, called Erasure Nodes, are capable to release full slots that carried information to upstream stations.

Several methods to incorporate slot reuse efficiently into the original DQDB MAC have been suggested in the past [5], [6], [7], [8], [9]. However, some of them can lead to starvation. In this paper we improve the scheme of [5] and the main contribution of the paper is that for the first time we also prove that the improved scheme guarantees lack of starvation.

The rest of the paper is organized as follows: In Section 2 we describe the original scheme of [5] that we later improve in Section 3. In Section 4 we introduce some notation which will help us later to simplify our proofs. In Section 5 we prove that our improved scheme guarantees lack of starvation and in Section 6 we present simulation results that demonstrate the efficiency of our improved scheme. Finally, in Appendix A we specify the proof for lack of starvation for the important case of DQDB without slot reuse.

2 The 'cancelled requests' scheme for slot reuse

The description of the scheme is based on [5]. It is implemented in every station that performs slot reuse/release either if it is an Erasure Node or in the case of Destination Release. The observation behind this scheme is that a released slot can be used by the releasing station either to satisfy a request from downstream stations or to be used for a self transmission. For efficiency reasons upstream stations shall be notified about this satisfaction since they can reduce their obligations to downstream stations. On the other hand, if a station releases a slot when it does not have requests from downstream stations or packets to transmit, it is not clear if the slot will be reused after its release. Therefore, a station maintain an additional counter, the ES (Erased Slot) counter. ES counts the number of released slots that were used to satisfy requests from downstream stations or for self transmissions. Every such released slot
The configuration consists of two unidirectional buses to which every station is connected by a pair of a transmitter (T) and a receiver (R). Stations use Bus A to transmit to stations with higher indices and Bus B to transmit to stations with lower indices.

The transmissions are done by slots. For each bus there is a slot generator that produces a stream of empty slots that travel along the bus until they reach the terminator where they are taken out. The slot generator is a part of the most upstream station of each bus.

DQDB is a MAC designed for the Dual Bus configuration. It enables a full utilization of the transmission medium, independent of its length or its transmission speed. In DQDB every slot contains several control bits which are called the ACF - Access Control Field. One control bit is the B (Busy) bit which indicates whether the slot has already been used for transmission (B=1) or not (B=0). Stations can transmit in empty slots only, turning the B bit from 0 to 1.

Clearly, if any station would have been allowed to transmit in any passing empty slot, this could lead to starvation at downstream stations when upstream stations are heavily loaded. Therefore, DQDB employs a special mechanism to guarantee lack of starvation. This mechanism employs another bit in the ACF, the R (Request) bit, which is generated with a '0' value. Also, two sets of three counters in every node, CD, RQ and REQ are used, one set for each bus. In the following we concentrate on transmissions on Bus A only. The transmissions on Bus B are done in a symmetric way.

When a station I has no packet for transmission, it increments RQ with every passing set R bit on Bus B, decrements RQ with every passing empty slot on Bus A and if REQ>0 it tries to set any unset R bit (R=0) passing on Bus B, decrementing REQ when succeeding in doing so.

When I has a packet for transmission, it sets CD←RQ, RQ←0 and increments REQ. I now increments RQ with every set R bit arriving on Bus B and decrements CD with every passing empty slot on Bus A. When CD reaches 0, I transmits its packet in the next empty slot arriving on Bus A. The operations concerning the REQ counter proceed as before. Only after transmitting a packet, I is allowed to schedule another packet for transmission, repeating the above procedure again. In Appendix A we prove that the original DQDB ensures lack of starvation.

\[1\text{In this paper we assume one priority only.}\]
A Proof for lack of starvation in DQDB with and without Slot Reuse

Oran Sharon
Dept. of Computer Science
Technion, Israel Institute of Technology
Haifa, Israel 32000

oran@csc.cs.technion.ac.il

Abstract

The DQDB is the IEEE 802.6 MAC standard for Metropolitan Area Networks. It is designed to operate on the Slotted, Dual Bus configuration. A simple way to improve the performance of this MAC is to perform slot reuse either by Destination Release or by Erasure Nodes, and many suggestions have been presented in the past to incorporate the slot reuse efficiently into the original DQDB MAC. However, some of these suggestions can lead to starvation. In this paper we improve one of these previous suggestions and the main contribution of the paper is that for the first time we also provide a proof that the improved scheme ensures lack of starvation. We also specify this proof for the important case of DQDB without slot reuse.

1 Introduction

The Dual Bus configuration is a popular configuration for LANs and MANs and several MACs have been designed for this topology [1], [2], [3], [4]. The topology is shown in Figure 1.