BLOCKING ANALYSIS OF TIME-DRIVEN SWITCHING

Viet-Thang Nguyen, Renato Lo Cigno, Yoram Ofek

December 2006

Technical Report # DIT-06-092
Blocking Analysis of Time-Driven Switching

Viet-Thang Nguyen, Renato Lo Cigno, Yoram Ofek
Dep. of Information and Communication Technology - University of Trento
Via Sommarive, 14 I-38050, Povo - Trento - Italy
Email: {nguyen,locigno,ofek}@dit.unitn.it

Abstract

This technical report presents a complete closed-form time-domain analysis of the blocking probability of time-driven switching (TDS) for the single node case. In this work blocking is defined as the occurrence in which transmission resources are available in both inlet and outlet, but there is no schedule. The main constraints for finding a schedule are: (i) the load and (ii) the maximum scheduling delay between inlet availability and outlet availability. As the maximum scheduling delay (buffering) increases the blocking probability is reduced. The outcome of the analysis in this report is the exact blocking probabilities for all possible maximum scheduling delays, under all load conditions.

I. INTRODUCTION

This technical report focuses on the single switch complete blocking probability analysis of time-driven switching (TDS). In the following we first introduce TDS, as discussed in Section I-A. Section I-B presents a more general discussion on the blocking issue, while in Section I-C examines some related works on blocking analysis. The remaining parts of the report are organized in the following way. Section II formulates the time-blocking problem, then Section III presents the overall approach used to compute the time-blocking probability. Section IV analyzes the blocking problem for the least complex case when there is only one buffer per inlet (for delaying the content of an incoming time-frame for the duration of zero or one time-frame - more details in the next subsection). The solution for general cases is then presented in Section V. Finally, Section VI presents some concluding discussion and future works.

A. Time-Driven Switching

Time-Driven Switching [3]–[6] is a network architecture that is based on the use of global time-of-day or UTC (coordinated universal time, a.k.a. Greenwich Mean Time - GMT). The principle underlying
TDS networks is pipeline forwarding (PF), a method known to provide optimal performance independent of specific implementation. The necessary condition for pipeline forwarding is having a common time reference, which in the context of this work is global time-of-day or UTC with proper accuracy. Note that TDS requires phase synchronization (i.e., time-of-day), which is entirely different than the very accurate frequency synchronization required by SONET/SDH. Note that what is required for TDS is that the time at any two points around the globe will be within maximum deviation of a few microseconds, which is not a demanding requirement and can be easily obtained from GPS or Galileo and for a low cost.

Specifically, TDS utilizes the UTC second that is partitioned into a predefined number of time-frames (time-frames). time-frames can be viewed as virtual containers for multiple variable-length IP packets that are switched as a whole at every TDS switch. The manners in which IP packets within time-frames are switched from inlet to outlet depend on UTC. Namely, for every time-frame within the UTC second there is a well defined switch configuration (i.e., inlet/outlet permutation), which does not drift in time, and consequently, enables the following non-trivial closed-form analysis.

A group of \( K \) time-frames forms a time-cycle (TC); \( L \) contiguous time-cycles are grouped into a super cycle that is equal to one UTC second, as shown in Fig.1. In this example: \( K = 1000 \) and \( L = 80 \). In TDS, all time-frames are aligned with UTC at the inlet ports prior to switching. After alignment, the delay between inlets of any pair of switches is an integer number of time-frames, which is the necessary condition for pipeline forwarding.

Fig. 1. Division of an UTC second in TDS.

TDS defines two types of time-frame forwarding with the corresponding maximum scheduling delays:
Immediate forwarding (IF): — upon the arrival of each time-frame to a TDS switch, the content of
time-frame (i.e, IP packets) is scheduled to be “immediately” switched and forwarded to the next switch,
as shown in Fig.2. Hence, excluding the alignment delay and the propagation delay, IF requires zero
scheduling delay.

Non-immediate forwarding (NIF): — which requires that the content of time-frame would be delayed
one or more time-frames at the TDS switch (i.e., non-zero scheduling delay). Let us assume that, at each
switch inlet there is a buffer for $z$ time-frames. (Note that each buffer can be either an optical delay line
or a solid state memory). Thus, the content of each time-frame arriving to the TDS switch can be buffered
for an arbitrary number $k_z$ time-frames ($0 \leq k_z \leq z$) before being forwarded to the next switch, as shown
in Fig.2, consequently, the maximum scheduling delay is $z$ time-frame durations. (Note that NIF does not
exclude IF.)

In essence, our analysis objective is to compute the blocking probability as a function of $z$. Note that
the special case $z = 0$ was analyzed in [6], while in this manuscript we analyze the general case of:
$1 \leq z \leq K - 1$. Note that for the case $z = K - 1$ the blocking probability is zero, simply because if
there is an inlet available time-frame and an outlet available time-frame, the maximum delay between
them cannot be greater than $K - 1$ time-frames. Indeed, this is what is shown in the following analysis.

B. Space and Time Blocking

In TDS, when a single switch is analyzed, there are two basic blocking issues: blocking in the space
domain (space-blocking); and blocking in the time domain (time-blocking).

Intuitively, if there are schedulable time-frames (e.g., free time-frames that satisfy any chosen forwarding
scheme) at a pair of inlet and outlet but the switch cannot be configured to form a forwarding path through
the switch fabric (i.e., no available resource in the switch fabric), it is defined as a space-blocking. Space-
blocking depends on the architecture of the switching fabric. Naturally, space-blocking can be completely
avoided by the deployment of strictly non-space-blocking fabrics [6].

On the other hand, even if there is no space-blocking there still can be blocking in the time domain.
For instance, consider the IF scheme when there are free time-frames at both an inlet and an outlet of a
switch, but the available (free) time-frames are having different time index within the time cycle, then
those free time-frames cannot be used for IF and there is blocking. In this case, we call it time-blocking.

The time-blocking is intrinsic in TDS and can be reduced by using buffers for flexible scheduling
as in the NIF schemes. Intuitively, NIF offers a greater flexibility in scheduling time-frames at TDS
switches, thus resulting in better performances regarding time-blocking probability. For example, assume
that time-frame 5 within the TC is available at the inlet and time-frame 7 within the TC is available at the outlet, then with two buffers (or scheduling delay of two time-frames) it is possible to forward the IP packets within time-frame 5 to the outlet at time-frame 7.

This work focuses on a quantitative time-blocking probability analysis. The time-blocking probability analysis in this work is a novel combinatorial approach. The main assumption is that all possible load combinations are equally likely. Namely, if a combination is defined by the distribution of $b$ busy time-frames (out of $K$ possible time-frames in each time cycle - TC) in a given inlet and a given outlet, then all such possible combinations are equally likely.

C. Related Works

Blocking performance analysis has its long history since conventional public phone networks were starting to be deployed [2]. Traditionally, the term ‘call blocking’ was used in previous works on blocking analysis (e.g., in [7]–[9]). ‘Call rejection’ is considered as an event when no more network resources (e.g., circuits in telephony or radio channels in wireless) can be allocated to successfully establish a newly arrival call. Thus, an analysis of ‘call rejection’ probability is called ‘call blocking’ probability analysis. When analyzing ‘call blocking’ probability, traffic patterns and stochastic distributions are taken into accounted.

However, in this work, we do not study the blocking probability at the call level. A blocking in the time-domain (formally defined in Section II) occurs even when there are available network resources (i.e., available time-frames) at both inlet and outlet of a TDS switch. A time-blocking occurs not due to running out of resources, but because no schedule can be found to properly allocate available resources (i.e., free time-frames).

II. Formal Problem Statement

The switch is part of a large network, we assume independence of each channel (i.e., inlet and outlet), thus we can examine a single channel of the switch. Since the traffic loading the channel comes from other nodes, the resources it uses are defined by the other nodes’ constraints, and cannot be assigned freely by the considered node. Assuming independence between nodes, we come to the following model for the traffic load.

Load assumptions — The load is defined as the number of busy time-frames per TC per channel. For all channels, the busy time-frames within each TC is assumed to be distributed uniformly. Let $b$ denotes
the number of busy time-frames per TC. The load of a channel is identified by the pair \((K, b)\) and it is further assumed that all possible combinations are equally likely.

It is further postulated that \(i\) the number \(b\) is identical for all inlets and outlets, and \(ii\) the distribution is independent, i.e., the time-frame distribution of the inlet is independent from the one of the outlet. The later assumption is rather restrictive for small switches, but can be reasonable for large ones.

To formulate the problem, we further define some notations:

- \(a\) denotes the number of free time-frames per TC, \(a = K - b\).
- \(t_{f_k}\) denotes a generic time-frame \(k\) in a TC.
- \(t_{f_{k_{in}}}\) denotes time-frame \(k\) of the inlet, \(0 \leq k < K\).
- \(t_{f_{k_{out}}}\) denotes time-frame \(k\) of the outlet, \(0 \leq k < K\).
- \(z\) denotes the number of buffers (or maximum scheduling delay), \(0 \leq z < K\).
- Symbol ‘0’ presents a busy time-frame.
- Symbol ‘1’ presents an available (or free) time-frame.

For all definitions and descriptions, note that a time-frame index has periodic attribute. In other words, if \(k \geq K\) then \(k = (k \mod K)\) since \(K\) time-frames are grouped in a TC.

**Def. 1:** \(z\)-forwarding scheme — A switch is said to be under \(z\)-forwarding scheme iff a content of a time-frame, upon its arrival, can be buffered arbitrarily for amount of \(i\) time-frame durations prior to being forwarded, \(i = 0, 1, \ldots, z\).

In other words, the maximum scheduling delay of an arrival time-frame is \(z\) time-frame durations under \(z\)-forwarding scheme. Note that \(z = 0\) means the immediate-forwarding (IF) scheme or zero scheduling delay.

**Def. 2:** \(z\)-schedulable time-frame — For a pair of inlet and outlet, \(t_{f_{k_{in}}}\) is said to be \(z\)-schedulable iff it is free and at least one time-frame in the set \(\{t_{f_{k+i_{out}}} | i = 0, 1, \ldots, z\}\) is free.

**Def. 3:** \(z\)-blocked time-frame — For a pair of inlet and outlet, \(t_{f_{k_{in}}}\) is said to be \(z\)-blocked iff it is free and all time-frames in the set \(\{t_{f_{k+i_{out}}} | i = 0, 1, \ldots, z\}\) are not free.

Examples of \(z\)-schedulable and \(z\)-blocked time-frames are in Fig.4.
**Problem statement** — Regarding a pair of inlet and outlet of a strictly non space-blocking switch operating under \( z \)-forwarding scheme, we aim at deriving the probability \( p_{xf} \), the time-blocking probability that all free time-frames of the inlet are found \( z \)-blocked (toward the outlet), given the load specified by \((K, b)\).

Let \( C_{blk} \) be the number of combinations made by both the inlet and the outlet such that all the \( a \) free time-frames of the inlet are found \( z \)-blocked. Let \( C_{total} \) be the total number of combinations made by both the inlet and the outlet. The time-blocking probability is the ratio between \( C_{blk} \) and \( C_{total} \):

\[
p_{xf} = \frac{C_{blk}}{C_{total}}
\]

The time-blocking probability for the IF scheme \((z = 0)\) was shown in [6]:

\[
p_b = 1 - p_a = 1 - \sum_{o=b}^{a=b} \binom{b}{o} \binom{K-b}{b-o} / \binom{K}{b}
\]

where we first computed the probability that there is at least one time-frame is found schedulable \( p_a \). Then the blocking probability was obtained as \( p_b = 1 - p_a \).

In this work, we present a direct computation of the time-blocking probability for all \( z \)-forwarding schemes by deriving combinatorial numbers \( C_{blk} \) and \( C_{total} \).

### III. Analysis Methodology

**A. Run, Run-length and Blocked Positions**

1) **Run and run-length:** Mainly, the discussion is focused on different dispositions of the \( a \) symbols ‘1’ and the \( b \) symbols ‘0’ in the outlet. A run is defined as a group of the same symbols that are positioned consecutively. For examples, runs of 0’s are ‘0’, ‘00’, ‘000’ and so on.

A number of symbols composing a run is its run-length. The minimum run-length for a meaningful run is 1. In between two adjacent runs of 0’s there is one run of 1’s, and vice versa.

![Illustration when \( K = 18, a = 4, z = 2 \): \( tf^{1^n} \) is \( z \)-blocked; \( tf^{0^o}, tf^{0^o} \) and \( tf^{1^o} \) are \( z \)-schedulable.](image)

Fig. 4. Illustration when \( K = 18, a = 4, z = 2 \): \( tf^{1^n} \) is \( z \)-blocked; \( tf^{0^o}, tf^{0^o} \) and \( tf^{1^o} \) are \( z \)-schedulable.
2) Runs in a cyclical arrangement: Because of the periodic nature of TDS, the last time-frame in a TC, for example, can be delayed until first time-frame positions in the next TC if \( z > 0 \). It means that the last time-frame and the first time-frame are positioned consecutively. This implies that the arrangement of the \( a \) symbols ‘1’ and the \( b \) symbols ‘0’ into \( K \) time-frame positions reoccur in a cyclical manner. Therefore, in each arrangement the number of runs of 0’s and the number of runs of 1’s are equal, excluding the trivial cases of all zeros and all ones.

![Example of runs in a cycle permutation.](image)

For instance, in a cyclical arrangement of 4 symbols ‘1’ and 14 symbols ‘0’, shown in Fig.5, there are 4 runs of 1’s and 4 runs of 0’s. One special run of 0’s whose run-length is 5 (3 time-frame positions at the beginning of the TC and 2 time-frame positions at the end of the TC).

3) Runs in a linear arrangement: In this case it is assumed that the cycle is open, and therefore, in Fig.5, under the linear arrangement view, there are 5 runs of 0’s and 4 runs of 1’s.

Note that all notations for runs and run-lengths presented in this report are defined for cyclical arrangements. However, in some parts of the combinatorial analysis, it is clearly indicated that the counting is of runs under the linear arrangement (e.g., in Section IV-A). We also note that for all linear arrangements discussed, the cycle is broken at the first time-frame position \( t_{f_0} \).

4) Blocked positions: Observe that for a given \( z \)-forwarding scheme, an arrangement of the \( a \) free time-frames and the \( b \) busy time-frames, in the outlet, generates some positions, such that if a free time-frame in the inlet is “positioned” above anyone of these positions, it is \( z \)-blocked. Thus, such positions are called blocked positions. In order to highlight the concept of blocked position, which is important in the following analysis, let’s consider the following examples:

- For \( z = 0 \) (i.e., the IF scheme), any arrangement in the outlet generates \( b \) blocked positions. Obviously, if a free inlet’s time-frame is ‘positioned above’ a busy outlet’s time-frame, it is blocked since \( z = 0 \).
- For \( z = 1 \), a content of time-frame can be delayed at most one time-frame duration prior to being forwarded. Fig.6 shows how blocked positions are generated. In fact, for every pair of adjacent ‘0 0’ symbols the left symbol generates a blocked position. Consequently, if there are \( l \) consecutive 0’s, then there are \( l - 1 \) blocked positions.
- For \( z = 2 \), a content of time-frame can be delayed at most two time-frame durations. Fig.7 shows
Fig. 6. Illustration of blocked positions when \( z = 1 \), given a sample combination of the outlet.

Fig. 7. Illustration of blocked positions when \( z = 2 \), given a combination of the outlet.

how blocked positions are generated in this case. Only runs whose run-length is greater than two (since \( z = 2 \)), such as, ‘000’, ‘0000’ and so on, generate blocked positions. Consequently, if there are \( l \) consecutive 0’s and \( l > 2 \), then there are \( l - 2 \) blocked positions.

From above illustrations, it is trivial to conclude that:

- The number of blocked positions generated in a given arrangement (of free time-frames and busy time-frames) in the outlet depends on a specific \( z \)-forwarding scheme and a given load \((K, b)\).
- For a run of 0’s, there is a relation between the number of blocked positions generated, its run-length and \( z \). Let \( l_i \) be the run-length of run \( i \) of 0’s. Let \( r_i \) be the number of blocked positions generated by run \( i \), then:

\[
r_i = l_i - z
\]

Therefore, we are interested in run \( i \) such that:

\[
l_i \geq z. \tag{4}
\]

5) The bound of the number of blocked positions: Given an arrangement in the outlet, let \( r \) be the total number of blocked positions generated from all runs of 0’s, the following is shown.

**Lemma 1:** For a given load \((K, b)\), \( r \) is bounded by:

\[
b - za = r_{\text{min}} \leq r \leq r_{\text{max}} = b - z \tag{5}
\]

**Proof:** From (3), we yield \( r_{\text{max}} = b - z \) when all the \( b \) ‘0’ symbols form a single run in the outlet, which is obviously the longest possible run. To compute \( r_{\text{min}} \), we further observe that, in a cyclical arrangement, \( a \) symbols of ‘1’ can split maximum \( a \) runs of 0’s, where every run has the same length of
z (i.e., \( l_i = z \) for all \( i \)) such that no blocked position is generated according to (3). The remaining number of ‘0’ symbols is \((b - za)\). Since no more run of 0’s can be formed due to running out of ‘1’ symbols to split them. Thus, placements of remaining ‘0’ symbols finally generate blocked positions. Therefore, 

\[ r_{\min} = (b - za). \]

The result in (5) is used to derive the generic form of time-blocking probability in subsection III-B.

The blocked position concept and the definition of \( z \)-blocked time-frame (Def. 3) imply that the time-blocking case happens when all free time-frames of the inlet are ‘placed’ in blocked positions. In the example shown in Fig.7, all free time-frames of the inlet \((t_f^\text{in}_k \text{ for } k = 1, 2, 13, 15)\) are blocked. Thus, we define each of these incidents as a time-blocking case.

**B. The General Form Of The Time-Blocking Probability**

For a given value of \( r \) satisfying (5), let \( C(r) \) be the number of arrangements found only in the outlet such that each of these arrangements generate exactly \( r \) blocked positions. We derive \( C(r) \) later in Section IV (for \( z = 1 \)) and V (for the general case). (In essence, counting \( C(r) \) is the most difficult part of our analysis and it is done in the subsequent two sections.) Given \( C(r) \), we have the following result:

**Theorem 1:** The time-blocking probability for the general \( z \)-forwarding scheme, \( p_{xF} \) is computed by:

\[
p_{xF} = \sum_{r = \max\{a, (b - za)\}}^{b - z} \frac{C(r)}{\binom{K}{b}^2}
\]

**Proof:** Given \( r \) blocked positions in the outlet, the number of ways to arrange all \( a \) free time-frames of the inlet into blocked positions so that a time-blocking occurs is \( \binom{r}{a} \). Thus, the subtotal number of combinations, denoted as \( C_{\text{sub}} \), generated by both the inlet and the outlet such that a time-blocking happens is given by:

\[
C_{\text{sub}} = C(r) \binom{r}{a}
\]

Note that if \( r < a \), then \( \binom{r}{a} = 0 \). Thus, we only consider \( r \geq a \) (i.e., a case where a time-blocking occurs). From Lemma 1, observe that:

- if \((b - za) \geq a \leftrightarrow K \geq (z + 2)a\) then for any combination in the outlet, we have \( r_{\min} = (b - za) \geq a \).
- if \((b - za) < a \leftrightarrow K < (z + 2)a\) then for some \( r \) such that \( b - za \leq r < a \), we are not interested in. Thus we set \( r_{\min} = a \).

Combined with (5) we have the range of meaningful \( r \) for computing time-blocking probability:

\[
\max\{a, (b - za)\} \leq r \leq b - z.
\]
The sum of $C_{\text{sub}}$ over all meaningful $r$ yields $C_{\text{blk}}$:

$$C_{\text{blk}} = \sum_{r = r_{\text{min}}}^{r_{\text{max}}} C_{\text{sub}} = \sum_{r = \max\{a,(b-z)\}}^{b-z} C_{(r)} \binom{r}{a}$$

Meanwhile, total numbers of combinations at the inlet and at the outlet are computed as $\binom{K}{b}$ for each inlet and outlet. Thus, we have $C_{\text{total}}$:

$$C_{\text{total}} = \binom{K}{b} \binom{K}{b} = \binom{K}{b}^2$$

Therefore, we obtain $p_{zF}$ as in (6) according to (1).

Theorem 1 shows how $p_{zF}$ is computed once we have $C_{(r)}$. However, the most nontrivial task is at the derivation of $C_{(r)}$, the number of combinations in the outlet generating exactly $r$ blocked positions. The computation is more complicated for $z$-forwarding schemes such that $z > 1$. Thus, in the next section, we first derive $C_{(r)}$ for $z = 1$. The derivation of $C_{(r)}$ for the general $z$-forwarding case is presented in Section V.

IV. THE 1-FORWARDING SCHEME

We separate the analysis of the 1-forwarding scheme from the general case, because its simpler mathematics allows for descriptions and explanations that will help in deriving the general case. 1-forwarding means there is a single position in the buffer: $z = 1$.

Let $u$ denote the number of runs of 0’s. For $z = 1$ all runs satisfy (4). Summing eq. (3) over all runs yields:

$$\sum_{i=1}^{u} r_i = \sum_{i=1}^{u} l_i - z = \sum_{i=1}^{u} l_i - uz$$

Since $\sum_{i=1}^{u} r_i = r$ (total number of blocked positions) and $\sum_{i=1}^{u} l_i = b$ (total number of symbols ‘0’), the equation above becomes simply:

$$u = b - r$$

Eq. (8) holds only for $z = 1$, and it is the reason why this case can be treated differently from the general one. In this case the computation of $C_{(r)}$ can be done in two different ways. The first one, considering a linear disposition of the symbols, gives the result with a problem decomposition in form of summation. The second one, which will be used also in the general case, considers the cyclic disposition of the symbols and gives the results in form of a multiplicative decomposition that, however, counts the number of possible patterns $u$ times, so that the final result must be divided by $u$. 
A. Additive Decomposition

In a non-cyclic perspective, the patterns into which the $b$ symbols ‘0’ and the $a$ symbols ‘1’ in an outlet can be disposed falls into one of the following cases:

Case 1: the first and the last symbol of the cycle are different, implying that there are $u$ runs of 0’s and $u$ runs of 1’s. The disposition in the outlet where $u = 4$ in Fig.6 falls into this case. Case 1 has two obvious and identical (from the combinatorial point of view) sub-cases: the first symbol is ‘0’ and the last one is ‘1’, or viceversa.

Case 2: both the first and the last symbol of the cycle are ‘1’ so that there are $u$ runs of 0’s, and $(u+1)$ runs of 1’s.

Case 3: both the first and the last symbol of the cycle are ‘0’, so that there are $(u+1)$ runs of 0’s and $u$ runs of 1’s.

It is easy to see that the three cases above form a partition of the set of the dispositions, and this is valid for any given $r$, so that $C_{(r)}$ can be computed as the sum of the three cases.

**Lemma 2:** For $z = 1$, $C_{(r)}$ is:

$$C_{(r)} = \frac{K}{u} \left( \frac{a-1}{u-1} \right) \left( \frac{b-1}{u-1} \right)$$

where $r$ is implicit in $u$ as in (8).

**Proof:** We sum all the combinations of the three cases defined above, that, forming a partition, contain all and only the distributions of interest, i.e.,

$$C_{(r)} = C_{\text{case 1}} + C_{\text{case 2}} + C_{\text{case 3}}$$

Consider case 1. The number of dispositions is the product of the following terms:

- the number of dispositions of the $a$ symbols ‘1’ into $u$ distinct runs such that there will be at least one symbol per run. Basic combinatorics (see [1] C.2) yields $\binom{a-1}{u-1}$.

- the number of dispositions of the $b$ symbols ‘0’ into $u$ distinct runs such that there will be at least one symbol per run, which is $\binom{b-1}{u-1}$.

- a multiplicative factor of 2 reporting of the two subcases.

Thus, we obtain $C_{\text{case 1}}$:

$$C_{\text{case 1}} = 2 \left( \frac{a-1}{u-1} \right) \left( \frac{b-1}{u-1} \right)$$

Following the same counting methods we obtain:

$$C_{\text{case 2}} = \left( \frac{a-1}{u} \right) \left( \frac{b-1}{u-1} \right) = \frac{a-u}{u} \left( \frac{a-1}{u-1} \right) \left( \frac{b-1}{u-1} \right)$$
\[ C_{\text{case 3}} = \binom{a - 1}{u - 1} \binom{b - 1}{u} = \frac{b - u}{u} \binom{a - 1}{u - 1} \binom{b - 1}{u - 1} \]

Summing together the three cases leads to eq. (9) with trivial algebra.

Substitute (9) into (6), \( u = b - r \), \( z = 1 \) and \( a = K - b \), we yield the time-blocking probability for the 1-forwarding scheme, \( p_{1F} \):

\[
p_{1F} = \sum_{r=\max\{a,(b-a)\}}^{b-1} \frac{K}{b} \binom{K-b-1}{b-r-1} \binom{b-1}{r} \binom{r}{b}^2 \tag{10}
\]

Fig. 8 shows numerical examples obtained from (10) and (2). In the graph, numerical results for \( K = 64, z = 1 \) and for \( K = 128, z = 0 \) are very close to each other. However, a quick investigation on the actual numbers shows that they are not identical, but only very similar. This can be explained as following. Once one buffer \( z = 1 \) is used, the effect is as comparable as we double both the number of free and the number of busy time-frames in a cycle and not use the buffer \( z = 0 \). Meanwhile, the reverse (i.e., moving from \( K = 128 \) to \( K = 64 \) and using \( z = 1 \)) does not hold.

**B. Direct Factorization**

Considering the space (i.e., the cycle) where the time-frames are disposed in a circle, where the last time-frame is adjacent to the first time-frame, a direct factorization of the counting problem is possible, counting all the possible dispositions of the runs of ‘1’ and ‘0’, however this leads to counting all the patterns \( u \) times, so that the final result must be divided by \( u \). Since this is the technique we use in the general case, we do not repeat it here, but refer to the next section.
V. The General $z$-Forwarding Scheme

Eq. (8) holds only for $z = 1$, since this is the only case where all the runs of 0’s satisfy (4). If (8) is not valid, there is not a unique relationship between $r$, $u$ and $b$, and the scenario becomes more complex.

When (4) is not satisfied by all runs of 0’s, these runs are divided into two subsets: those that lead to blocking positions and those that do not. Let’s introduce the following notations, that will be used in deriving the results later in the section:

- $U$ denotes the set of all runs $i$ (of 0’s) such that run-lengths satisfy (4). Only runs with $l_i > z$ produces $r_i \geq 1$, i.e., blocking positions.
- $u = |U|$.
- $b_u$ denotes the number of symbols ‘0’ covered by all runs in $U$.
- $V$ denotes the set of all runs $i$ (of 0’s) such that run-length $l_i < z$. That is, no run in $V$ produces blocked positions.
- $v = |V|$.
- $b_v$ denotes the number of symbols ‘0’ covered by all runs in $V$.
- $A$ denotes the set of all runs of 1’s. Thus, $u + v = |A|$.

From the above definitions it is immediately clear that the 1-forwarding case is the special case where $V = \emptyset$. Table I summarizes the notations introduced above, together with other ones already used in the rest of the report.

One of the key differences between the 1-forwarding case and the general case analyzed is the presence of non-valid $(u, v)$ couples, i.e., values of $u$ and $v$ that do not satisfy all the constraints of the problem. This fact forces us to separately count for all and any the valid $(u, v)$ couples, while the simple relation (8) allowed for a unique computation. Given this additional complexity, partitioning the set of patterns as we did for $z = 1$ becomes excessively cumbersome, so we resort to the analysis considering the cyclic disposition of time-frames.

We now define some general bounds for the parameters of the problem, that will be the upper and lower limits of the indexes used in the formulae derived afterwards. Summing eq. (3) over all runs in $U$ yields (with some simple algebra manipulations):

$$0 < b_u = r + zu \leq b$$  \hspace{1cm} (11)

The number of symbols $b_v$ is given by:

$$b_v = b - b_u = b - r - zu \geq 0$$  \hspace{1cm} (12)
TABLE I

SUMMARY OF THE NOTATION USED FOR THE GENERAL CASE

<table>
<thead>
<tr>
<th>Notation</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha$</td>
<td>Number of symbols ‘1’ (i.e. number of free time-frames)</td>
</tr>
<tr>
<td>$b$</td>
<td>Number of symbols ‘0’ (i.e. number of busy time-frames)</td>
</tr>
<tr>
<td>$z$</td>
<td>Number of buffers, $1 \leq z &lt; K$</td>
</tr>
<tr>
<td>$p_{z,f}$</td>
<td>Blocking probability under $z$-forwarding scheme</td>
</tr>
<tr>
<td>$l_i$</td>
<td>Run-length of run $i$</td>
</tr>
<tr>
<td>$r_i$</td>
<td>Number of blocked positions generated by run $i$</td>
</tr>
<tr>
<td>$r$</td>
<td>Total number of blocked positions generated by all runs of 0’s in a given arrangement</td>
</tr>
<tr>
<td>$U$</td>
<td>Set of all runs of 0’s such that $l_i \geq z$</td>
</tr>
<tr>
<td>$u$</td>
<td>Number of runs in $U$, $u = |U|$</td>
</tr>
<tr>
<td>$b_u$</td>
<td>Total number of symbols ‘0’ occupied by all runs in $U$</td>
</tr>
<tr>
<td>$V$</td>
<td>Set of all runs of 0’s such that $1 \leq l_i &lt; z$</td>
</tr>
<tr>
<td>$v$</td>
<td>Number of runs in $V$, $v = |V|$</td>
</tr>
<tr>
<td>$b_v$</td>
<td>Total number of symbols ‘0’ occupied by all runs in $V$</td>
</tr>
<tr>
<td>$A$</td>
<td>Set of all runs of 1’s, $u + v = |A|$</td>
</tr>
<tr>
<td>$C_{(u,v)}$</td>
<td>Number of combinations that generate exact $r$ blocked positions, given a valid pair of $(u,v)$</td>
</tr>
<tr>
<td>$C_{(r)}$</td>
<td>Total number of combinations in the outlet that generates exact $r$ blocked positions, for all valid pairs of $(u,v)$</td>
</tr>
</tbody>
</table>

While by construction, we have:

$$1 \leq u + v \leq \alpha$$  \hspace{1cm} (13)

**Lemma 3:** The size of $U$ is bounded by:

$$1 = u_{min} \leq u \leq u_{max} = \min\{\left\lfloor \frac{b - r}{z} \right\rfloor, a\}$$  \hspace{1cm} (14)

**Proof:** When there is only one run of 0’s, we have $u_{min} = 1$.

From (11) we have $u = \frac{b - r}{z}$ and $u = u_{max}$ iff $b_u = b$. $b_u = b$ implies that all symbols ‘0’ of the outlet are in runs belonging to $U$ and $V=\emptyset$, $v = 0$. Setting $v = 0$ in (13) yields $u \leq a$ so that $u_{max} \leq \min\{\left\lfloor \frac{b - r}{z} \right\rfloor, a\}$. 
Note that $u = 0$ is not considered since it means there is one run of 0’s with length smaller than $z$, or $b < z$. In this case we do not have time-blocking.

**Lemma 4:** For $1 < z < K$, the size of $\mathbb{V}$ is bounded by:

$$\left\lfloor \frac{b_v}{z-1} \right\rfloor = v_{\text{min}} \leq v \leq v_{\text{max}} = \min\{a - u, b_v\} \quad (15)$$

**Proof:** We have $v = v_{\text{min}} = \left\lfloor \frac{b_v}{z-1} \right\rfloor$ when all runs in $\mathbb{V}$ have the maximum allowed length $l_i = (z-1)$.

The upper bound depends on the ratio between $b_v$ and the number of symbols ‘1’ not used to separate runs in $\mathbb{V}$ that can separate runs in $\mathbb{U}$. That is $(a - u)$.

- if $b_v > (a - u)$ then $v_{\text{max}} = (a - u)$.
- if $b_v \leq (a - u)$, we can split all $b_v$ symbols ‘0’ in runs of length one, so that $v_{\text{max}} = b_v$.

Therefore, $v_{\text{max}} = \min\{a - u, b_v\}$. ■

A. Deriving The Combinatorial Number $C_{(r)}$

Eqs. (11)-(15) define the limits of $(u, v)$ for a given value of blocking positions $r$ satisfying (7).

Recall that in a time-cycle, the last time-frame $tf_{K-1}$ is considered to be adjacent to (on the left) the first time-frame $tf_0$, so that no real “beginning” or “ending” of the cycle exist and no run is “split” as it happens considering the linear disposition with the cycle’s beginning and ending.

**Theorem 2:** Given a valid pair of $(u, v)$, the number of patterns, denoted as $C_{(u,v)}$, that exactly generate $r$ blocked positions is:

$$C_{(u,v)} = \frac{KC_{uv}C_aC_{bu}C_{bv}}{u + v} \quad (16)$$

where $r$ is implicit in $b_u, b_v, u, v$ given the relations (11)-(15). The factors $C_{uv}$, $C_a$, $C_{bu}$, and $C_{bv}$ are defined in (17)-(20) of the proof, respectively.

**Proof:** The goal is computing the total number of possible patterns distributing the $b_u$ symbols ‘0’ into $\mathbb{U}$ runs, the $b_v$ symbols ‘0’ into $\mathbb{V}$ runs, and the $a$ symbols ‘1’ into runs in $\mathbb{A}$. To obtain this we show that there exists a factorization of the problem that counts $(u+v)$ times the total number of patterns. The factorization starts counting the possible dispositions of the runs themselves given $u$ and $v$, then counts the dispositions of the symbols in the runs in different sets $\mathbb{A}$, $\mathbb{U}$, and $\mathbb{V}$, finally all possible $K$ cyclic shifts of the above patterns are counted showing that each pattern is thus counted exactly $(u+v)$ times.

$C_{uv}$: number of dispositions of the $u$ runs in $\mathbb{U}$ within the total number of possible runs $(u+v)$ of $\mathbb{U} \cup \mathbb{V}$. Trivial combinatorics yields:

$$C_{uv} = \binom{u + v}{u} \quad (17)$$
$C_a$: number of dispositions of the $a$ symbols ‘1’ into the $(u + v)$ distinct runs such that each run has at least one symbol. Basic combinatorics [1] yields:

$$C_a = \binom{a - 1}{u + v - 1}$$  \hfill (18)

$C_u$: number of dispositions of the $b_u$ symbols ‘0’ into the $u$ distinct runs such that each run has at least $z$ symbols. The counting method consists in first placing $(z - 1)$ symbols into every run $\in U$, then distributing the remaining $b_u - (z - 1)u$ symbols in all the $u$ runs such that each run has at least one symbol. Using the same combinatoric result used for $C_a$ we have:

$$C_{b_u} = \binom{b_u - (z - 1)u - 1}{u - 1}$$  \hfill (19)

$C_{b_v}$: number dispositions of the $b_v$ symbols ‘0’ into the $v$ distinct runs such that each run has at least one symbol and no run has more than $(z - 1)$ symbols:

$$C_{b_v} = \begin{cases} \sum_{i=0}^{v} (-1)^i \binom{v}{i} \binom{b_v - i(z - 1) - 1}{v - 1}, & v > 0 \\ 1, & v = 0 \text{ or } b_v = v \end{cases}$$  \hfill (20a)

$$C_{b_v} = \begin{cases} \sum_{i=0}^{v} (-1)^i \binom{v}{i} \binom{b_v - i(z - 1) - 1}{v - 1}, & v > 0 \\ 1, & v = 0 \text{ or } b_v = v \end{cases}$$  \hfill (20b)

Deriving (20) is a little cumbersome and we present it in Appendix A.

The time-cycle boundary can be at any time-frame, thus there are $K$ possible shifts for each disposition counted so far. The total number of possible dispositions given a valid pair $(u, v)$ is then $KC_{uv}C_aC_{b_u}C_{b_v}$. However, each combination is actually counted $(u + v)$ times and the number $KC_{uv}C_aC_{b_u}C_{b_v}$ must be divided by $(u + v)$ to eliminate multiple countings, thus resulting in (16).

The proof of the multiple counting is presented in Appendix B to streamline reading. The rationale is that each of the $C_{uv}C_aC_{b_u}C_{b_v}$ can be transformed into exactly $u + v$ other patterns by shifting it circularly of an appropriate number of time-frames.

**Theorem 3:** The total number of dispositions $C_{(r)}$ that generates exact $r$ blocked positions is given by:

$$C_{(r)} = \sum_{u=1}^{\min\{\frac{u+v}v, a\}} \left\{ \sum_{v=\lceil \frac{b_v}u \rceil}^{\min\{u, b_v\}} C_{(u,v)} \big|_{uv \leq a} \right\}$$  \hfill (21)

**Proof:** A pair of $(u, v)$ is valid iff $u$ and $v$ jointly satisfy (13), (14) and (15). Since $C_{(u,v)}$ is computed through eq. (16) for any valid pair of $(u, v)$, the sum of $C_{(u,v)}$ over all valid pairs of $(u, v)$ leads to the total number of dispositions $C_{(r)}$ in the outlet that generates exact $r$ blocked positions.

Finally, summing (21) over all valid values of $r$ fulfills the numerator of (6) and finally the closed form solution of the time-blocking probability.
Examples of numerical results for various $z$ and $K$ values are shown in Fig.11. One interesting property is the reduction in time-blocking probability as $K$ increases for a given normalized load. While it was an easy prediction that time-blocking probability would decrease exponentially increasing the buffering
Fig. 11. Numerical results: $K = 256$, $z$ varies.

capability $z$, a similar decrease simply increasing $K$ was not an easy prediction. The phenomenon is similar to the classic result that gives smaller and smaller call blocking probability for a given load as the granularity of the calls decreases.

B. Sanity Checks

The result for the general case presented above is rather complex and might be appalling. Here we discuss some limit cases where the exact result can be easily obtained with heuristic reasoning.

1) For $gF_iK$:

This is the case of immediate forwarding. For any combination in the outlet, we always have $v = r = r_{min} = r_{max} = b$, and $C(r) = \binom{K}{b}$. Thus, the eq. (6) shrinks to:

$$p_{OF} = \frac{\binom{K}{b}(b)}{\binom{K}{b}^2} = \frac{b}{\binom{K}{b}}$$

(22)

Numerical results of (2) and of (22) are coincident.

2) For $gF_iR$:

For $z = 1$, we have $V = \emptyset$ or $b_v = 0$, and $b_u = b = r + u$. Letting $v = 0$ in the formulae of theorem 2 yields

$$C_{(u,v=0)} = \frac{K}{u} \left( \frac{a - 1}{u - 1} \right) \left( \frac{b_u - (z - 1)u - 1}{u - 1} \right)$$

(23)
which, as claimed in Section IV-B, is equivalent to (9) remembering that \( z = 1 \Leftrightarrow b = b_u; \ v = 0; \ r = b - u. \)

3) For \( b \leq K - 1, \ z = K - 1: \)

Replacing \( z = K - 1 \) into (5) yields \( r_{max} = b - z = b - (K - 1) \leq 0 \) since \( b \leq K - 1 \). This implies that there is no single combination where we can find \( r \geq a \), or \( p_{zf} = 0 \) for \( z = K - 1 \). The intuition of zero blocking for this special case is confirmed.

VI. DISCUSSION AND FUTURE WORKS

The problem of time-blocking probability in TDS switches has been formulated and fully analyzed in this work. It has been shown that time-blocking is greatly reduced when a small number of (optical) buffers (used for enabling scheduling delays that are measured in time-frames) are added to each inlet. Consequently, the main result of this report is the time-blocking probability analysis as a function of the number of buffers \( z \) and the load \( (K, b) \).

The time-blocking probability analysis in this work is a novel combinatorial approach. The main assumption is that all possible load combinations are equally likely. Namely, if a combination is defined by the distribution on \( b \) busy time-frames, out of \( K \) possible time-frames in each time-cycle, in a given inlet and a given outlet, then all such possible combinations are equally likely. Some concrete numerical examples were presented in the report, which clearly illustrating that only a small number of buffers (or short scheduling delay) is required to obtain low blocking probability under high load conditions.

However, the analysis presented in this work is suitable to comprehend time-blocking behaviors only for a single strictly non space-blocking switch. Thus, we are currently extending the blocking analysis work in the following non-trivial directions:

1) Multiple hops with IF \( (z = 0) \) and NIF \( (z > 0) \).

2) Multiple hops with WDM (for fractional lambda switching - F\( \lambda \)S [6]) and with IF and NIF.

3) Space-blocking fabric (e.g., multi-stage Banyan) with WDM and with IF and NIF.

APPENDIX

A. Proof of (20)

First, let us find the number of ways \( C(m, n, p) \) to put \( m \) identical balls into \( n \) distinct boxes such that no box has more than \( p \) balls, \( p > 1 \).

While this may look like a classic combinatorial problem, we failed to find appropriate references also outside the field of blocking probabilities. The above problem is equivalent to finding the number of
integer solutions to the equation:

\[ e_1 + e_2 + e_3 + \ldots + e_n = m \quad \quad 0 \leq e_i \leq p \]

The generating function for the above equation is:

\[ h(x) = (1 + x + x^2 + x^3 + x^4 \ldots + x^p)^n \]

The problem turns to finding the coefficient of \( x^m \) in the polynomial \( h(x) \).

From a well-known polynomial identity

\[ \frac{1 - x^{p+1}}{1 - x} = 1 + x + x^2 + x^3 + \ldots + x^p \]

we can represent \( h(x) = f(x)g(x) \) where \( f(x) = \frac{1}{(1-x)^n} \) and \( g(x) = (1 - x^{p+1})^n \).

Following are polynomial expansions in chapter 3 of [1]:

\[ f(x) = \frac{1}{(1-x)^n} = \sum_{i=0}^{\infty} \alpha_i x^i \]

where

\[ \alpha_i = \binom{i + n - 1}{i} \]

and

\[ g(x) = (1 - x^{p+1})^n = \sum_{i=0}^{n} \beta_{i(p+1)} x^{i(p+1)} \]

where

\[ \beta_{i(p+1)} = \begin{cases} (-1)^i \binom{n}{i}, & \text{for } i = 0, 1, \ldots, n \\ 0, & \text{otherwise} \end{cases} \]

Therefore, \( h(x) \) can be rewritten as \( h(x) = \sum_{q=0}^{\infty} \Lambda_q x^q \) where the coefficients \( \Lambda_q \) are given by:

\[ \Lambda_q = \sum_{i=0}^{q} \alpha_{q-i} \beta_i \] (24)

We aim at finding the coefficient of \( x^m \) in \( h(x) \), thus we only need to consider the terms \( \alpha_{m-i} \beta_i \) in which the \( \beta_i \)'s, coefficients of \( g(x) \), are nonzero. Substitute \( q = m \) into (24), we have:

\[ C(m, n, p) = \Lambda_m = \sum_{i=0}^{n} \binom{-i(p+1)+n-1}{m-i(p+1)} (-1)^i \binom{n}{i} \]

\[ = \sum_{i=0}^{n} (-1)^i \binom{n}{i} \binom{m-i(p+1)+n-1}{n-1} \] (25)

Substituting \( m = b_v - v \), \( n = v \), and \( p = z - 2 \) into (25), we obtain \( C_{b_v} \) as in (20) for \( v > 0 \).

Besides, notice that:

- if \( v = 0 \), we set \( C_{b_v} = 1 \) since \( C_{b_v} \) is a factor of a product.
- if \( b_v = v \) then obviously \( C_{b_v} = 1 \).
B. Proof of multiple counting while deriving (16)

Let $\Sigma$ denote the set of all combinations generated in the product $C_uC_aC_bC_v$.

Consider one pattern $\chi_1$ drawn at random from $\Sigma$, and label all runs of $\chi_1$ in an increasing order within their proper set as following:

$$\chi_1 = u_1a_1v_1a_2\cdots u_i\cdots a_j\cdots v_k\cdots u_k\cdots v_\alpha a_{u+v}$$

in which $u_i \in U$, $v_k \in V$, and $a_j \in A$; $i = 1, 2, \cdots, u$; $k = 1, 2, \cdots, v$; and $j = 1, 2, \cdots, u+v$.

By construction, also all the following patterns are items of the set $\Sigma$ and they are distinct:

$$\chi_2 = v_1a_2\cdots u_i\cdots a_j\cdots v_k\cdots v_\alpha a_{u+v}u_1a_1$$
$$\vdots$$
$$\chi_{u+v} = v_\alpha a_{u+v}u_1a_1v_1a_2\cdots u_i\cdots a_j\cdots v_k\cdots$$

It is clear that $\chi_2, \cdots, \chi_{u+v}$ are also obtained as $s$-position shifts of $\chi_1$, with proper $s < K$.

More formally, letting $\chi_i^s$ denote a left shifting on $\chi_i$ with shifting length $s$, and $|*_i|$ length of a generic run $*_i$, we have:

$$\chi_2^{|u_1|+|a_1|} \equiv \chi_1$$
$$\vdots \equiv \chi_1$$
$$\chi_{u+v}^{|u_1|+|a_1|+|v_1|+|a_2|+\cdots+|u_i|+\cdots+|a_j|+\cdots+|v_k|+\cdots} \equiv \chi_1$$

Since $\chi_1$ was selected randomly, we conclude that any $\chi_i$ appears exactly $(u + v)$ times in the total number $KC_uC_aC_bC_v$.

REFERENCES

