Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.18576/amis/140103

Modeling of Network Packet Switches Using Matrix Analysis

Oladele Theophilus Sule and Roberto Rojas-Cessa*

Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102, USA

Received: 5 Mar. 2019, Revised: 28 Jun. 2019, Accepted: 14 Aug. 2019
Published online: 1 Jan. 2020

Abstract: We introduce the use of matrix analysis as a method for describing the operation of a packet switch in this paper. Matrix analysis may be used to statistically describe how incoming traffic is switched from inputs to outputs according to the architecture of the switch and the selection of packets for forwarding. This method finds a direct application on throughput analysis of a packet switch, defined by the switch architecture and configuration scheme. We show the applicability of this method on different known switch architectures, as examples, and show that the results are consistent with their reported performance in the literature. Specifically, the method is applied to single- and multi-stage switches using load-balancing or matching and arbitration schemes in their configuration process. Moreover, the method is used to obtain the throughput of a switch and its corresponding configuration scheme for a given doubly stochastic input matrix.

Keywords: Matrix decomposition, performance analysis, throughput analysis, switch operation

1 Introduction

The operation of a packet switch (or just switch for brevity) is to forward Internet packets from inputs to outputs. An $N \times N$ switch may have its inputs indexed by $0 \leq u \leq N-1$ and its outputs by $0 \leq v \leq N-1$. Different packet switch architectures require different configuration schemes. These schemes have a direct effect on the operation of switching traffic from inputs to outputs and, therefore, on the performance of the switch. Performing switch analysis is a critical tool to which designers resort to determine the worthiness of a switch architecture. Moreover, these tools can be specifically targeted at identifying a performance metric, such as stability, throughput, or delay among the most demanded. However, analyzing the different performance metrics of the switch also requires different tools, some of which sometimes come at the expense of high complexity or render complex to apply.

Here, we consider that a packet switch may receive variable length Internet packets and segment them into fixed-size cells for internal switching, and re-assembling them before packets depart the switch. In this way, the time it takes to switch a cell from inputs to outputs is also fixed, and it is referred to a time slot.

The performance of a switch may be determined by different metrics. Some of the more sought after are:

Throughput. In general, throughput of a switch is conveniently represented as a normalized estimate; that is, the ratio of the number of cells that left the outputs of the switch to the arrived cells at the inputs of the switch. The throughput of a switch is one the the most important performance metrics as it enables us to evaluate other parameters, such a delay. The delay of a cell (or packet) can be evaluated only if the throughput of a switch for a given input load is full; that is, the number of cells leaving the switch equals that entering the switch. The throughput of a switch may be estimated by computer simulation, probabilistic analysis, direct estimation, or tight or loose bound estimation for cases where absolute estimation is complex [1–9]. In the particular case 100% throughput is expected under 100% input load (full load that inputs may sustain), stability of the queues, and therefore, the switch may be analyzed as binary test [10–18].

Stability. A switch is considered to be stable if for any admissible input traffic, the switch is capable of forwarding the incoming traffic to the respective output(s), such that the queue occupancy of the switch does not grow infinitely.

* Corresponding author e-mail: ots5@njit.edu

We define admissible traffic as:

$$\sum_{u=0}^{N-1} \lambda_{u,v} \leq 1, \sum_{v=0}^{N-1} \lambda_{u,v} \leq 1 \quad (1)$$

where $\lambda_{u,v}$ is the arrival rate of traffic from input u to output v of a switch. Such admissibility is considered here under independent and identically-distributed (i.i.d.) traffic conditions.

In this case, the queue occupancy is not expected to grow if the input is admissible. However, some queueing may occur for short periods of time but as long as the average profile of the input load remains admissible, queues are not expected to grow infinitely, if input buffers have sufficient capacity [12–15]. The finite queue occupancy of a switch can be evaluated by computer simulation [19], modeling the queueing system of the switch [20], by using a fluid model to determine its stability [11], or by using a stability criteria such as Lyapunov analysis. For a switch with queue occupancy matrix $\mathbf{L}(n)$ at time n, state vector

$$Y(n) = (\mathbf{L}(n), X(n)) \quad (2)$$

where $X(n)$ is an integer vector, and a Lyapunov function

$$V(L(n)) = L(n)\, Z\, L(n)^T \quad (3)$$

If there is a symmetric copositive, an $N \times N$ matrix B is copositive if $L\, \mathrm{B}\, L^T \geq 0\ \forall\ L \in \mathbb{R}^{+N}$ matrix $Z \in \mathbb{R}^{N \times N}$, and two positive numbers $\epsilon \in \mathbb{R}^+$ and $U \in \mathbb{R}^+$, such that:

$$E[V(L(n+1) - V(L(n)|L(n)] < -\epsilon ||L(n)||\ \ \forall\, Y(n) :$$

$$||L(n)|| > U \quad (4)$$

the switch is considered to be stable [12–15, 21–24].

Queueing delay. The queueing delay, or just delay, is the time a cell waits in a queue before being forwarded to the destination output. This waiting time may depend on both the presence of other cells in other queues contending to be forwarded to the output and on the scheme used to select the next cell to be forwarded. Then, the average queueing delay of a switch is the average estimate on the cells passing through the switch. For analysis of a switch under i.i.d. traffic, the average switch delay of switch is equivalent to that experienced by traffic coming through one of the outputs.

The average queueing delay of a switch may be measured through computer simulation or by modeling the queues as a Markov process model and analyzing the model, or queueing analysis [2, 20, 22, 23, 25–29]. However, the last option is difficult to adapt for various switch architectures, traffic types, and selection schemes.

These issues raise the question, can the throughput of a switch be analyzed using a simpler method that provides insight into the switching fabric and numeric throughput values?

We propose the use of matrix analysis as a tool for analyzing the operation of a switch on the incoming traffic and thus, to analyze the performance of a switch. Matrix analysis provides a deep insight into the operation of the switching fabric, and also with this analysis we may obtain a numeric throughput.

The use of matrix decomposition as a configuration scheme, has been previously applied to a few switching architectures. In [30], a matrix decomposition algorithm was presented to route cells in a rearrangeable three-stage Clos network by performing a row-wise matrix decomposition. In [31, 32], a scheduling algorithm that uses the results from Birkhoff decomposition [33] and von Neumann algorithms [34] for a doubly substochastic matrix was proposed for an input buffered crossbar switch.

Different from those applications, our objective is to apply matrix analysis in modeling the operation of the switch on the incoming traffic and to estimate the performance of the switch. In this paper, we show that matrix analysis can be applied to both single and multi-stage switch architectures that use either pre-determined or random arbitration schemes.

The remainder of this paper is organized as follows: Section 2 introduces our matrix analysis approach in general terms. Section 3 describes several application cases, from single to multi-stage switches that may use load-balancing functions or other configuration schemes. In this examples, we estimate the switch throughput or verify that a switch achieves 100% throughput. Section 4 presents our conclusion.

2 Throughput Analysis through Matrix Analysis

For any single or multistage switching architecture the internal configuration of the switch can be represented by a compound permutation $\mathbf{P}$, or a set of permutations $\mathbf{P_1}, \cdots, \mathbf{P}_{\gamma-1}$. where $\gamma > 1$ and it is the number of stages in the multistage architecture.

Let's consider that the traffic incoming to a switch, or input matrix $\mathbf{R_1}$, is defined as:

$$\mathbf{R_1} = [\lambda_{u,v}] \quad (5)$$

where $\mathbf{R_1}$ is admissible.

The traffic at the output port of a single-stage switch, $\mathbf{R_2}$, is:

$$\mathbf{R_2} = \mathbf{R_1} \alpha \mathbf{P} \quad (6)$$

where α is defined by the configuration of the switch and $\mathbf{P}$ is the connection provided by the configuration. Moreover, for a multistage switch architecture with γ stages, the matrix at the output port, $\mathbf{R}_\gamma$, is:

$$\mathbf{R}_\gamma = (((\mathbf{R_1} \alpha \mathbf{P_1}) \alpha \mathbf{P_2}) \cdots \alpha \mathbf{P}_{\gamma-1}) \quad (7)$$

The throughput of a single-stage or multi-stage switch with input $\mathbf{R_1}$ and output $\mathbf{R_2}$ or $\mathbf{R}_\gamma$, respectively, is calculated by dividing the sum of the columns in $\mathbf{R_2}$ or $\mathbf{R}_\gamma$ by the sum of the columns in $\mathbf{R_1}$. For an output matrix, $\mathbf{R_O}$, or:

$$\mathbf{R_O} = [\beta_{\mu,v}] \tag{8}$$

where $\mathbf{R_O} = \mathbf{R_2}$ for a single-stage, $\mathbf{R_O} = \mathbf{R}_\gamma$ for a multi-stage switch, and $\beta_{\mu,v}$ is the traffic arrival rate to output port v of the switch from an input μ. The throughput of the switch is calculated by:

$$\text{Throughput} = \frac{\sum_{v=0}^{N-1}\sum_{\mu=0}^{N-1}\beta_{\mu,v}}{\sum_{v=0}^{N-1}\sum_{u=0}^{N-1}\lambda_{u,v}} \tag{9}$$

3 Application Examples of Matrix Analysis on Switch Architectures

In this Section, we apply this technique to an output queued (OQ), input-queued, and load-balancing Birkhoff-von-Neumann (LBvN) switches, which can be classified as single (e.g., OQ and IQ) and multi-stages switches (e.g., LBvN and MM^{e}M), with deterministic (e.g., LBvN) and non-deterministic configuration patterns. These switches are selected because of their high performance and herein, we show that the proposed approach can be used to estimate the switch performance. The architecture of the switches used in the examples are shown in Figure 1.

3.1 Output-queued (OQ) Switch

In this section we analyze the performance of an output-queued (OQ) switch. An OQ switch has queues at the outputs. Figure 1(a) shows the architecture of an OQ switch.

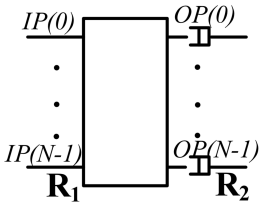
Let us denote the traffic coming to the input ports and then leaving to the output ports of the OQ switch as $\mathbf{R_1}$, $\mathbf{R_2}$, respectively. Here, $\mathbf{R_1}$ and $\mathbf{R_2}$ are $N \times N$ matrices.

Here, $\mathbf{R_1}$ is obtained from (5), the configuration of the OQ switching fabric at time slot t that connects $IP(i)$ to $OP(j)$ is represented as an $N \times N$ permutation matrix, $\mathbf{\Pi}(t) = [\pi_{u,v}]$, and the matrix element:

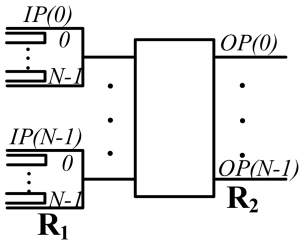
$$\pi_{u,v} = \begin{cases} 1 & \text{if } u \text{ connects to } v \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

The configuration of the OQ switch can be represented as a compound permutation matrix, $\mathbf{P}$, as follows,

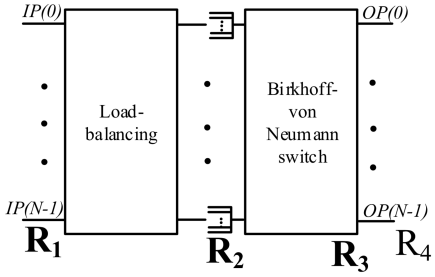
$$\mathbf{P} = \sum_t \mathbf{\Pi}(t) \tag{11}$$



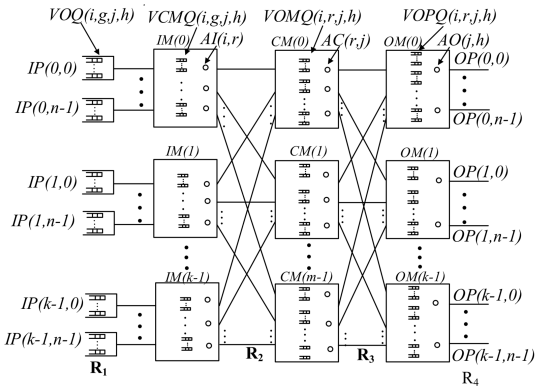
(a) OQ switch



(b) IQ switch



(c) LBvN switch



(d) MM^{e}M switch

Fig. 1: Switch architectures of example switches.

While $\mathbf{R_2}$ is:

$$\mathbf{R_2} = (\mathbf{R_1} \circ \mathbf{P}) \tag{12}$$

where $\circ$ denotes element- and position-wise multiplication.

As an example, we calculate the throughput for a 4×4 OQ switch for any input traffic pattern. Let the input traffic matrix be:

$$\mathbf{R_1} = \begin{bmatrix} \lambda_{0,0} & \lambda_{0,1} & \lambda_{0,2} & \lambda_{0,3} \\ \lambda_{1,0} & \lambda_{1,1} & \lambda_{1,2} & \lambda_{1,3} \\ \lambda_{2,0} & \lambda_{2,1} & \lambda_{2,2} & \lambda_{2,3} \\ \lambda_{3,0} & \lambda_{3,1} & \lambda_{3,2} & \lambda_{3,3} \end{bmatrix}$$

From (11), the compound permutation matrix of the OQ switch is:

$$\mathbf{P} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

Using (12), we get $\mathbf{R_2}$, the traffic matrix at the output ports:

$$\mathbf{R_2} = \begin{bmatrix} \lambda_{0,0} & \lambda_{0,1} & \lambda_{0,2} & \lambda_{0,3} \\ \lambda_{1,0} & \lambda_{1,1} & \lambda_{1,2} & \lambda_{1,3} \\ \lambda_{2,0} & \lambda_{2,1} & \lambda_{2,2} & \lambda_{2,3} \\ \lambda_{3,0} & \lambda_{3,1} & \lambda_{3,2} & \lambda_{3,3} \end{bmatrix} \tag{13}$$

From (13), we see that $\mathbf{R_1} = \mathbf{R_2}$ and this is achieved over one time slot, which implies a speedup of N. By using (9), it is easy to see that the OQ switch attains 100% throughput under admissible traffic and this result is equivalent to the well-known performance of this switch.

3.2 Input-queued Switch with iterative Round Robin Matching (iRRM)

In this section, we analyze an input queue (IQ) switch which uses iterative round robin matching (iRRM) as the configuration scheme [35]. An IQ switch has queues at the inputs and usually virtual output queues (VOQs) which are used to queue cells for each destination output. Figure 1(b) shows the architecture of an IQ switch. iRRM works as follows: at time slot t, all the unmatched IPs with one or more queued cells for an OP sends a request to the OP.

The OP chooses the request that appears next in a round-robin schedule starting from the OP with the highest priority, and notifies each requesting IP whether the request is granted. The round-robin pointer of the OP is updated to one location beyond the granted IP (modulo u).

The IP accepts the grant from an OP in a round-robin schedule starting from the OP with the highest priority and updates the round-robin pointer to one location beyond the accepted OP (modulo v).

Let us denote the traffic coming to the input and the output ports of the IQ switch as $\mathbf{R_1}$, $\mathbf{R_2}$, respectively, as $N \times N$ matrices.

The traffic matrix at the input port is obtained from (5), the configuration of the IQ switch at time t can be represented as an $N \times N$ permutation matrix, $\mathbf{\Pi}(t) = [\pi_{u,v}]$, and the matrix element:

$$\pi_{u,v} = \begin{cases} 1 & \text{if } u \text{ connects to } v \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

Because each IP connects to an OP in a round-robin fashion under the presence of traffic for that OP at the IP, the configuration of the switch can be represented as a pre-deterministic sequence of periodic permutations.

$$\mathbf{P}(t) = \sum \mathbf{\Pi}(t) \tag{15}$$

A permutation then indicates the connections from an IP to the OP for which it has traffic. This permutation may change as a pattern, in a round-robin schedule, each time slot.

The traffic matrix at the output ports after each time slot t, $\mathbf{R_P}$(t), is:

$$\mathbf{R_P}(t) = \mathbf{R_P}(t-1) - \alpha_t \mathbf{P}(t) \tag{16}$$

where α_t is the smallest weight element of $\mathbf{R_P}(t)$ at non-zero positions of $\mathbf{P}(t)$. In (16), $\mathbf{R_P}(t = 0)$ is the input matrix $\mathbf{R_1}$. Therefore, the traffic matrix at the output ports after N time slots, or the matrix for the matched connections, $\mathbf{R_2}$, is:

$$\mathbf{R_2} = \mathbf{R_1} - \mathbf{R_P}(t = N) \tag{17}$$

where $\mathbf{R_P}(t = N)$ is the matrix of unmatched connections.

Example of an IQ Switch with iRRM under Uniform Traffic

In this example, we consider a uniformly-distributed input traffic for a 4×4 IQ switch. Let the input traffic matrix be:

$$\mathbf{R_1} = \begin{bmatrix} \lambda_{0,0} & \lambda_{0,1} & \lambda_{0,2} & \lambda_{0,3} \\ \lambda_{1,0} & \lambda_{1,1} & \lambda_{1,2} & \lambda_{1,3} \\ \lambda_{2,0} & \lambda_{2,1} & \lambda_{2,2} & \lambda_{2,3} \\ \lambda_{3,0} & \lambda_{3,1} & \lambda_{3,2} & \lambda_{3,3} \end{bmatrix}$$

$\mathbf{R_1}$ is decomposed into $\mathbf{R_1}(t)$ for each time slot t. From (15) and (16), we obtain:

$$\mathbf{R_P}(1) = \begin{bmatrix} 0 & \lambda_{0,1} & \lambda_{0,2} & \lambda_{0,3} \\ \lambda_{1,0} & 0 & \lambda_{1,2} & \lambda_{1,3} \\ \lambda_{2,0} & \lambda_{2,1} & 0 & \lambda_{2,3} \\ \lambda_{3,0} & \lambda_{3,1} & \lambda_{3,2} & 0 \end{bmatrix}$$

$$\mathbf{R_P}(2) = \begin{bmatrix} 0 & \lambda_{0,1} & \lambda_{0,2} & 0 \\ 0 & 0 & \lambda_{1,2} & \lambda_{1,3} \\ \lambda_{2,0} & 0 & 0 & \lambda_{2,3} \\ \lambda_{3,0} & \lambda_{3,1} & 0 & 0 \end{bmatrix}$$

$$\mathbf{R_P}(3) = \begin{bmatrix} 0 & \lambda_{0,1} & 0 & 0 \\ 0 & 0 & \lambda_{1,2} & 0 \\ 0 & 0 & 0 & \lambda_{2,3} \\ \lambda_{3,0} & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{R_P}(4) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

where $\mathbf{R_P}(4)$ is the matrix for unmatched connections, but by being a null matrix, it indicates that all connections are matched.

Using (17), we obtain $\mathbf{R_2}$, the traffic matrix at the output ports after N time slots, or:

$$\mathbf{R_2} = \begin{bmatrix} \lambda_{0,0} & \lambda_{0,1} & \lambda_{0,2} & \lambda_{0,3} \\ \lambda_{1,0} & \lambda_{1,1} & \lambda_{1,2} & \lambda_{1,3} \\ \lambda_{2,0} & \lambda_{2,1} & \lambda_{2,2} & \lambda_{2,3} \\ \lambda_{3,0} & \lambda_{3,1} & \lambda_{3,2} & \lambda_{3,3} \end{bmatrix} \tag{18}$$

As stated above, each IP connects to each OP once every N time slots. Because $\mathbf{R_1}$ is uniformly distributed, each IP forwards traffic at a rate of $\frac{1}{N}$, which for this example is represented as:

$$\mathbf{R_1} = \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix} \tag{19}$$

Therefore,

$$\mathbf{R_2} = \begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{bmatrix} \tag{20}$$

From (18) and (20), we see that $\mathbf{R_1} = \mathbf{R_2}$. By using (9), it is clear that this switch achieves 100% throughput under uniform traffic.

Example of an IQ Switch with iRRM under Nonuniform Traffic

In this example, we consider a nonuniform input matrix and show the throughput of IQ with iRRM. Let $\mathbf{R_1}$ be:

$$\mathbf{R_1} = \begin{bmatrix} 0.4 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.4 & 0.2 & 0.2 \\ 0.2 & 0.4 & 0.1 & 0.3 \\ 0.2 & 0 & 0.5 & 0.3 \end{bmatrix} \tag{21}$$

From (15) and (16), we obtain:

$$\mathbf{R_P}(1) = \begin{bmatrix} 0.3 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.3 & 0.2 & 0.2 \\ 0.2 & 0.4 & 0 & 0.3 \\ 0.2 & 0 & 0.5 & 0.2 \end{bmatrix}$$

$$\mathbf{R_P}(2) = \begin{bmatrix} 0.3 & 0.2 & 0.2 & 0 \\ 0 & 0.3 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0 & 0.3 \\ 0.2 & 0 & 0.3 & 0.2 \end{bmatrix}$$

$$\mathbf{R_P}(3) = \begin{bmatrix} 0.3 & 0.2 & 0 & 0 \\ 0 & 0.3 & 0.2 & 0 \\ 0 & 0.2 & 0 & 0.3 \\ 0.2 & 0 & 0.3 & 0.2 \end{bmatrix}$$

$$\mathbf{R_P}(4) = \begin{bmatrix} 0.3 & 0 & 0 & 0 \\ 0 & 0.3 & 0 & 0 \\ 0 & 0.2 & 0 & 0.1 \\ 0 & 0 & 0.3 & 0.2 \end{bmatrix}$$

$\mathbf{R_P}(4)$ is the matrix of unmatched connections.

From (17), we obtain $\mathbf{R_2}$, the traffic matrix at the output ports after N time slots, or the traffic for the matched connections is:

$$\mathbf{R_2} = \begin{bmatrix} 0.1 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.1 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.1 & 0.2 \\ 0.2 & 0 & 0.2 & 0.1 \end{bmatrix} \tag{22}$$

Using (9), we obtain for throughput by summing the columns of (22), and dividing each sum by the sum of the same column of $\mathbf{R_1}$, and summing the results, or:

$$\text{Throughput} = \frac{0.7 + 0.5 + 0.7 + 0.7}{4} = 0.65 \tag{23}$$

Therefore, the attained throughput is 65% under this nonuniform traffic and after N time slots.

3.3 Load Balanced Birkhoff-von Neumann (LBvN) Switch

We consider a Load Balanced Birkhoff-von Neumann (LBvN) switch [26, 36] in this section. This switch consists of two stages, the first stage load balances the input traffic by using periodic permutations in its configuration, and the second stage which is a Birkhoff-von Neumann input-queued switch, switches the load-balanced traffic toward the destinations. Figure 1(c) shows the architecture of LBvN and the traffic matrix representation of each stage.

Let us denote the traffic coming to the first stage, leaving the first stage, entering the second stage, and leaving the LBvN switch as $\mathbf{R_1}$, $\mathbf{R_2}$, $\mathbf{R_3}$, and $\mathbf{R_4}$, respectively. Here, $\mathbf{R_1}$, $\mathbf{R_2}$, and $\mathbf{R_3}$ are $N \times N$ matrices and $\mathbf{R_4}$ is an $N \times 1$ column vector.

The configuration of the first stage that connects input port, $IP(i)$, to the output of the first stage, or internal output port, $IOP(i)$, at time slot t is represented as an $N \times N$ permutation matrix, $\mathbf{\Pi_{BvN}}(t) = [\pi_{u,v}]$, and the matrix element:

$$\pi_{u,v} = \begin{cases} 1 & \text{for any } u,\ v = (u+t) \mod N \\ 0 & \text{elsewhere.} \end{cases} \quad (24)$$

The configuration of the first stage can be represented as a compound permutation matrix, $\mathbf{P}$, which is the sum of the permutations used on the first stage over N time slots, as follows:

$$\mathbf{P} = \sum^{N} \mathbf{\Pi_{BvN}}(t) \quad (25)$$

Because a switch configuration is repeated every N time slots, the traffic load from the same input going to each VOMQ is $\frac{1}{N}$ of $\mathbf{R_1}$. The traffic matrix $\mathbf{R_2}$ is obtained by using:

$$\mathbf{R_2} = \frac{1}{N}((\mathbf{R_1} * \mathbb{1}) \circ \mathbf{P}) \quad (26)$$

Given that

$$\mathbf{P} = \mathbb{1}$$

where $\mathbb{1}$ denotes an $N \times N$ unit matrix. Hence:

$$\mathbf{R_2} = \frac{1}{N}(\mathbf{R_1} * \mathbf{P}) \quad (27)$$

Here, $\mathbf{R_2}$ is the aggregate output traffic from the first stage destined to all OPs. This matrix can be further decomposed into N $N \times N$ matrices, $\mathbf{R_2}(v)$, each of which is the aggregate traffic at the input of the second stage destined to OP_v.

$$\mathbf{R_2} = \sum_{v=0}^{N-1} \mathbf{R_2}(v) \quad (28)$$

The second stage of this switch runs a sequence of periodic connection patterns that repeats every N time slots. Hence, the configuration of the second stage can be represented as a compound permutation matrix, $\mathbf{P}$, which is similar to the compound permutation used by the first stage of the switch. Therefore, the traffic forwarded to an OP is:

$$\mathbf{R_3}(v) = (\mathbf{R_2}(v) \circ \mathbf{P}) * \mathbf{1} \quad (29)$$

where $\mathbf{1}$ is an $N \times 1$ vector of all ones. The traffic forwarded to all OPs, $\mathbf{R_3}$, is:

$$\mathbf{R_3} = [\mathbf{R_3}(0), \cdots, \mathbf{R_3}(N-1)] \quad (30)$$

The traffic leaving an OP, $R_4(v)$ is:

$$R_4(v) = (\mathbf{1})^T * \mathbf{R_3}(v) \quad (31)$$

The traffic leaving all OPs, $\mathbf{R_4}$ is:

$$\mathbf{R_4} = [R_4(0), \cdots, R_4(N-1)]^T \quad (32)$$

Example of a LBvN Switch under Nonuniform Traffic

We show a 4×4 LBvN switch under a nonuniform input matrix and show the throughput of achieved throughput. From Section 3.2, it is easy to see that LBvN attains 100% throughput under uniform traffic. Let $\mathbf{R_1}$ be the same used in (21).

The input traffic into the second stage, $\mathbf{R_2}$, is generated from $\mathbf{R_1}$. From (25), the compound permutation matrix of the first stage is:

$$\mathbf{P_1} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

This permutation enables connecting each IP to OP and forwarding at a rate of $\frac{1}{N}$, or $\frac{1}{4}$. Using (27), we get $\mathbf{R_2}$, the input traffic matrix of the second stage:

$$\mathbf{R_2} = \frac{1}{4}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

From (28), the traffic matrices at the input of the second stage destined to OPs are:

$$\mathbf{R_2}(0) = \begin{bmatrix} 0.1 & 0.1 & 0.1 & 0.1 \\ 0.05 & 0.05 & 0.05 & 0.05 \\ 0.05 & 0.05 & 0.05 & 0.05 \\ 0.05 & 0.05 & 0.05 & 0.05 \end{bmatrix}$$

$$\mathbf{R_2}(1) = \begin{bmatrix} 0.05 & 0.05 & 0.05 & 0.05 \\ 0.05 & 0.05 & 0.05 & 0.05 \\ 0.025 & 0.025 & 0.025 & 0.025 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{R_2}(2) = \begin{bmatrix} 0.05 & 0.05 & 0.05 & 0.05 \\ 0.05 & 0.05 & 0.05 & 0.05 \\ 0.025 & 0.025 & 0.025 & 0.025 \\ 0.125 & 0.125 & 0.125 & 0.125 \end{bmatrix}$$

$$\mathbf{R_2}(3) = \begin{bmatrix} 0.05 & 0.05 & 0.05 & 0.05 \\ 0.05 & 0.05 & 0.05 & 0.05 \\ 0.075 & 0.075 & 0.075 & 0.075 \\ 0.075 & 0.075 & 0.075 & 0.075 \end{bmatrix}$$

The columns of $\mathbf{R_2}(v)$ are the traffic at the queues of the IP of the second stage (IIP), and the rows are the traffic from the IP of the first stage. Using (29), the traffic forwarded into the second stage destined for each OP, $\mathbf{R_3}(v)$ is:

$$\mathbf{R_3}(0) = \begin{bmatrix} 0.4 \\ 0.2 \\ 0.2 \\ 0.2 \end{bmatrix}$$

$$\mathbf{R_3}(1) = \begin{bmatrix} 0.2 \\ 0.4 \\ 0.4 \\ 0 \end{bmatrix}$$

$$\mathbf{R_3}(2) = \begin{bmatrix} 0.2 \\ 0.2 \\ 0.1 \\ 0.5 \end{bmatrix}$$

$$\mathbf{R_3}(3) = \begin{bmatrix} 0.2 \\ 0.2 \\ 0.3 \\ 0.3 \end{bmatrix}$$

The rows of $\mathbf{R_3}(v)$ represent the traffic from IPs forwarded through an IIP. Using (30), the aggregate traffic forwarded into the second stage, $\mathbf{R_3}$, is:

$$\mathbf{R_3} = \begin{bmatrix} 0.4 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.4 & 0.2 & 0.2 \\ 0.2 & 0.4 & 0.1 & 0.3 \\ 0.2 & 0 & 0.5 & 0.3 \end{bmatrix} \tag{33}$$

The traffic leaving each OP is obtained from (31):

$$R_4(0) = 1 \tag{34}$$

$$R_4(1) = 1 \tag{35}$$

$$R_4(2) = 1 \tag{36}$$

$$R_4(3) = 1 \tag{37}$$

The traffic leaving all OPs is obtained from (32):

$$\mathbf{R_4} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \tag{38}$$

Equation (33) is the output matrix, and by using (9) or from the output vector (38), it is easy to see that this switch achieves 100% throughput under nonuniform traffic. These results are consistent with those presented in [26, 36].

Non-blocking Memory-Memory-Memory with Extended Memory (MM^{e}M) Clos-Network Switch

Now let us focus on a multiple-stage switch as another example. This switch is the non-blocking memory-memory-memory with extended memory Clos-network switch (MM^{e}M) [37–39]. This switch is a three-stage buffered Clos-network architecture consisting of k $n \times m$ Input modules (IMs), k $m \times n$ Output modules (OMs), and m $p \times p$ Central modules (CMs). There are nN crosspoint buffers in an IM and OM, and kN crosspoint buffers in a CM. Each switch module has per-output flow queues to avoid HoL blocking. Round-robin or longest queue first arbitrations can be used in the IM, while round-robin is used in the CMs and OMs. The switch uses round-robin to arbitrate which queue forwards a cell to the output links of the modules of all three stages. Let us denote the traffic coming to the IPs, CMs, OPs, and the traffic leaving MM^{e}M as $\mathbf{R_1}$, $\mathbf{R_2}$, $\mathbf{R_3}$, and R_4 respectively. Here, $\mathbf{R_1}$, $\mathbf{R_2}$, and $\mathbf{R_3}$ are $N \times N$ matrices and $\mathbf{R_4}$ is an $N \times 1$ column vector.

Round-robin arbitration is used in IMs to select th IP permitted to forward a cell to the to the virtual central module queue ($VCMQ(i,s,r)$) at $IM(i)$ that stores cells from $IP(i,s)$ going through $CM(r)$ to its output port. This arbitration scheme enables forwarding a cell from $IP(i,s)$ to the queues in IM, and in turn forwarding cells from $IM(i)$ to output link $L_I(i,r)$ that connects the $IM(i)$ to $CM(r)$. This interconnection occurs once every n time slots for a fully-loaded switch. Similar to the selection of queues in IM, round-robin arbitration is used at a CM to select the per-output flow queue ($POFQ(i,r,j,d)$) that stores cells from $L_I(i,r)$ and destined to $OP(j,d)$. Figure 1(d) shows the architecture of MM^{e}M.

The specific configurations of the IM and CM are as follows. At time slot t, $IP(i,s)$ sends a cell to $VCMQ(i,s,r)$, and this VCMQ is connected to $L_I(i,r)$ in the following time slot, as follows:

$$r = (s+t) \mod m \tag{39}$$

Each CM input $L_I(i,r)$ is connected to $POFQ(i,r,j,d)$ and then this queue is connected to $L_C(r,j)$ as follows:

$$j = (r+t) \mod k \tag{40}$$

Then $\mathbf{R_2}$ is the traffic forwarded to the CMs and the product of having $\mathbf{R_1}$ switched by the permutations used in the configurations of IMs. The configuration of the IM stage at time slot t that interconnects $IP(i,s)$ to $VCMQ(i,r,j)$, and in turn interconnects to $L_I(i,r)$, are represented as an $N \times N$ permutation matrix, $\mathbf{\Pi_{Clos}}(t) = [\pi_{u,v}]$, where r is determined from (39) and the matrix element:

$$\pi_{u,v} = \begin{cases} 1 & \text{for any } u,\ v = rk + i \\ 0 & \text{otherwise.} \end{cases} \tag{41}$$

The configuration of the IM stage can be represented as a compound permutation matrix, $\mathbf{P_1}$, which is the sum of the IM permutations over m time slots as follows,

$$\mathbf{P_1} = \sum^{m} \mathbf{\Pi_{Clos}}(t) \tag{42}$$

Because the configuration repeats every m time slots, the traffic load from the same input going to each VCMQ is $\frac{1}{m}$

of the traffic load of $\mathbf{R_1}$. The traffic matrix $\mathbf{R_2}$ is obtained using:

$$\mathbf{R_2} = \frac{1}{m}((\mathbf{R_1} * \mathbb{1}) \circ \mathbf{P_1}) \tag{43}$$

Here, $\mathbf{R_2}$ is the aggregate traffic being forwarded to CMs destined to all OPs. This matrix can be further decomposed into N $N \times N$ submatrices, $\mathbf{R_2}(j,d)$, each of which is the aggregate traffic at CMs destined to $OP(j,d)$.

$$\mathbf{R_2} = \sum_{j=0}^{j=k-1} \sum_{d=0}^{d=n-1} \mathbf{R_2}(j,d) \tag{44}$$

The configuration of the CM stage at time slot t that connects $I_c(r,p)$ to $L_{C(r,j)}$ may be represented as an $N \times N$ permutation matrix, $\mathbf{\Phi}(t) = [\phi_{u,v}]$, where j is determined from (40) and the matrix element:

$$\phi_{u,v} = \begin{cases} 1 & \text{for any } u,\ v = jk + r \\ 0 & \text{otherwise.} \end{cases} \tag{45}$$

Similarly, the switching process at the CM stage is represented by a compound permutation matrix $\mathbf{P_2}$, which is the sum of p permutations of the CM stage over p time slots. Here,

$$\mathbf{P_2} = \sum^{p} \mathbf{\Phi}(t) \tag{46}$$

The traffic forwarded to the OMs and queued at the CB of an OP, $\mathbf{R_3}(j,d)$ is:

$$\mathbf{R_3}(j,d) = (\mathbf{R_2}(j,d) \circ \mathbf{P_2}) * \mathbf{1} \tag{47}$$

The aggregate traffic forwarded to the OMs, $\mathbf{R_3}$ is:

$$\mathbf{R_3} = [\mathbf{R_3}(0,0), \cdots, \mathbf{R_3}(k-1, n-1)] \tag{48}$$

The traffic leaving an OP, $R_4(j,d)$, is:

$$R_4(j,d) = (\mathbf{1})^T * \mathbf{R_3}(j,d) \tag{49}$$

The aggregate traffic leaving all OPs, $\mathbf{R_4}$, is:

$$\mathbf{R_4} = [R_4(0,0), \cdots, R_4(p-1, n-1)]^T \tag{50}$$

Example of a Buffered Clos-network Switch with Extended Memory (MM^{e}M) under Nonuniform Traffic

In this example, we consider a nonuniform input matrix as the traffic coming into MM^{e}M and calculate the switch throughput. We only consider nonuniform traffic in this example because from Section 3.2, it is easy to see that MM^{e}M attains 100% throughput under uniform traffic. Let $\mathbf{R_1}$ be the that used in (21).

From (42), the compound permutation matrix of the IM is:

$$\mathbf{P_1} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

This permutation enables connecting each IP to the output of the IM. Using (43), we get:

$$\mathbf{R_2} = 1/2 \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

From (44), the traffic matrix at the CMs destined for the different OPs are:

$$\mathbf{R_2}(0,0) = \begin{bmatrix} 0.2 & 0 & 0.2 & 0 \\ 0.1 & 0 & 0.1 & 0 \\ 0 & 0.1 & 0 & 0.1 \\ 0 & 0.1 & 0 & 0.1 \end{bmatrix}$$

$$\mathbf{R_2}(0,1) = \begin{bmatrix} 0.1 & 0 & 0.1 & 0 \\ 0.2 & 0 & 0.2 & 0 \\ 0 & 0.2 & 0 & 0.2 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{R_2}(1,0) = \begin{bmatrix} 0.1 & 0 & 0.1 & 0 \\ 0.1 & 0 & 0.1 & 0 \\ 0 & 0.05 & 0 & 0.05 \\ 0 & 0.25 & 0 & 0.25 \end{bmatrix}$$

$$\mathbf{R_2}(1,1) = \begin{bmatrix} 0.1 & 0 & 0.1 & 0 \\ 0.1 & 0 & 0.1 & 0 \\ 0 & 0.15 & 0 & 0.15 \\ 0 & 0.15 & 0 & 0.15 \end{bmatrix}$$

From (46), the compound permutation matrix for the CM stage for this switch is:

$$\mathbf{P_2} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

From (47), the traffic matrix forwarded to OMs and queued at the CBs of each OP, $\mathbf{R_3}(0,0)$ to $\mathbf{R_3}(1,1)$ are derived:

$$\mathbf{R_3}(0,0) = \begin{bmatrix} 0.4 \\ 0.2 \\ 0.2 \\ 0.2 \end{bmatrix}$$

$$\mathbf{R_3}(0,1) = \begin{bmatrix} 0.2 \\ 0.4 \\ 0.4 \\ 0 \end{bmatrix}$$

$$\mathbf{R_3}(1,0) = \begin{bmatrix} 0.2 \\ 0.2 \\ 0.1 \\ 0.5 \end{bmatrix}$$

$$\mathbf{R_3}(1,1) = \begin{bmatrix} 0.2 \\ 0.2 \\ 0.3 \\ 0.3 \end{bmatrix}$$

From (48), the traffic matrix forwarded to OMs and queued at the CBs of all OPs, $\mathbf{R_3}$ is:

$$\mathbf{R_3} = \begin{bmatrix} 0.4 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.4 & 0.2 & 0.2 \\ 0.2 & 0.4 & 0.1 & 0.3 \\ 0.2 & 0 & 0.5 & 0.3 \end{bmatrix} \tag{51}$$

Using (49), we obtain the traffic leaving each OP, or:

$$R_4(0) = 1 \tag{52}$$

$$R_4(1) = 1 \tag{53}$$

$$R_4(2) = 1 \tag{54}$$

$$R_4(3) = 1 \tag{55}$$

Using (50), we obtain the traffic leaving all OPs, or:

$$\mathbf{R_4} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \tag{56}$$

Equation (51) is the output matrix, and by using (9) or from the output vector (56), it is clear that this switch achieves 100% throughput under non-uniform traffic. These results are consistent with those presented in the original paper [37–39].

4 Conclusion

We have shown in this paper that the operation of a switch affects the average traffic coming into the switch and the overall performance of the switch. These operations can be represented as matrix operations, which may be used to estimate the throughput of a switch. We have shown the application of the proposed approach on examples of switch architectures with single or multiple stages, and with different queueing strategies and their corresponding configuration schemes. The results obtained in the presented examples are consistent with the known performance of such switches. By the examples provided, the tool is shown to be practical and can be used to analyze the performance of a wide set of packet switches.

References

[1] Li, Xin and Zhou, Zhen and Hamdi, Mounir, Space-memory-memory architecture for Clos-network packet switches, *Communications, 2005. ICC 2005. 2005 IEEE International Conference on*, IEEE **2**, 1031–1035 (2005).

[2] Jenq, Yih-Chyun, Performance analysis of a packet switch based on single-buffered banyan network, *IEEE Journal on Selected Areas in Communications* **6(1)**, 1014–1021 (1983).

[3] Dong, Ziqian and Rojas-Cessa, Roberto, Throughput analysis of shared-memory crosspoint buffered packet switches, *IET communications* **6(9)**, 1045–1053 (2012)

[4] Rojas-Cessa, Roberto and Dong, Ziqian, Load-balanced combined input-crosspoint buffered packet switches, *IEEE Transactions on Communications* **59(5)**, 1421–1433 (2011).

[5] Anderson, Thomas E and Owicki, Susan S and Saxe, James B and Thacker, Charles P,High-speed switch scheduling for local-area networks, *ACM Transactions on Computer Systems (TOCS)* **11(4)**, 319–352 (1993).

[6] Anderson, Thomas E and Owicki, Susan S and Saxe, James B and Thacker, Charles P, High speed switch scheduling for local area networks, *ACM SIGPLAN Notices* **27(9)**, 98–110 (1992).

[7] Kanizo, Yossi and Hay, David and Keslassy, Isaac, The crosspoint-queued switch, *INFOCOM 2009*, 729–737 (2009).

[8] Lin, Mingjie and McKeown, Nick, The throughput of a buffered crossbar switch, *IEEE communications Letters* **9(5)**, 465–467 (2005).

[9] Rojas-Cessa, Roberto and Lin, Chuan-bi, Matching schemes with captured-frame eligibility for input-queued packet switches, *Communications, 2005. ICC 2005. 2005 IEEE International Conference on* **2**, 972–976 (2005).

[10] Sule, Oladele Theophilus and Rojas-Cessa, Roberto and Dong, Ziqian and Lin, Chuan-Bi, A Split-Central-Buffered Load-Balancing Clos-Network Switch With In-Order Forwarding, *IEEE/ACM Transactions on Networking*, IEEE (2018).

[11] Dai, JG and Prabhakar, Balaji, The throughput of data switches with and without speedup, *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings*. IEEE **2**, 556–564 (2000).

[12] Mekkittikul, Adisak and McKeown, Nick, A starvation-free algorithm for achieving 100% throughput in an input-queued switch, *Proc. ICCCN* **96**, 226–231 (1996).

[13] McKeown, Nicholas William, *Scheduling algorithms for input-queued cell switches*, Ph.D. dissertation, University of California, Berkeley, (1995).

[14] Mekkittikul, Adisak and McKeown, Nick, Scheduling VOQ switches under non-uniform traffic, *CSL Technical Report, CSL-TR 97-747* (1997).

[15] Mekkittikul, Adisak and McKeown, Nick, A practical scheduling algorithm to achieve 100% throughput in input-queued switches, *INFOCOM'98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings*. IEEE **2**, 792–799 (1998).

[16] Hu, Bing and Yeung, Kwan L and Zhou, Qian and He, Chunzhi, On Iterative Scheduling for Input-Queued Switches With a Speedup of $2 - 1/N$, *IEEE/ACM Transactions on Networking* **24(6)**, 3565–3577 (2016).

[17] Rojas-Cessa, R and Guo, Zhen and Ansari, N, Combining distributed and centralized arbitration schemes for combined input-crosspoint buffered packet switches, *Networks, 2005. Jointly held with the 2005 IEEE 7th Malaysia International Conference on Communication., 2005 13th IEEE International Conference on* **2**, 776–770 (2005).

[18] Sule, Oladele Theophilus and Rojas-Cessa, Roberto, TRIDENT: A load-balancing Clos-network Packet Switch with Queues between Input and Central Stages and In-Order Forwarding, *IEEE Transactions on Communications* (2019).

[19] M. Lotfi and H. Mounir, CBF: A high-performance scheduling algorithm for buffered crossbar switches, *High Performance Switching and Routing, 2003, HPSR. Workshop on*, IEEE, 67–72 (2003).

[20] Rojas-Cessa, Roberto and Dong, Ziqian, Combined Input-Crosspoint Buffered Packet Switch with Flexible Access to Crosspoints Buffers, *Devices, Circuits and Systems, Proceedings of the 6th International Caribbean Conference on*, **21**, 255–260 (2006).

[21] Bastin, Georges and Haut, Bertrand and Coron, Jean-Michel and d Andrea-Novel, Brigitte, Lyapunov stability analysis of networks of scalar conservation laws, *Networks and Heterogeneous media* **4(2)**, 749 (2007).

[22] Shah, Devavrat and Kopikare, Milind, Delay bounds for approximate maximum weight matching algorithms for input queued switches, **2**, *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings*. IEEE, 1024–1031 (2002).

[23] Marsan, F Ajmone and Bianco, Andrea and Giaccone, Paolo and Leonardi, Emilio and Neri, E, Packet scheduling in input-queued cell-based switches, *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings*. IEEE **2**, 1085–1094 (2001).

[24] Rojas-Cessa, Roberto and Guo, Zhen and Ansari, Nirwan, On the maximum throughput of a combined input-crosspoint queued packet switch, *IEICE transactions on communications* **89(11)**, 3120–3123 (2006).

[25] Bertsekas, Dimitri P and Gallager, Robert G and Humblet, Pierre, *Data networks*, **2**, (1992).

[26] Chang, Cheng-Shang and Lee, Duan-Shin and Lien, Ching-Ming, Load balanced Birkhoff-von Neumann switches, part II: Multi-stage buffering.

[27] Iliadis, Ilias and Denzel, WOLFGANG E, Performance of packet switches with input and output queueing, *Communications, 1990. ICC'90, Including Supercomm Technical Sessions. SUPERCOMM/ICC'90. Conference Record., IEEE International Conference on*, **21**, 747–753 (1990).

[28] Pattavina, Achille and Bruzzi, Giacomo, Analysis of input and output queueing for nonblocking ATM switches, *IEEE/ACM Transactions on Networking (TON)* **1(3)**, 314–328 (1993).

[29] McKeown, Nick, The iSLIP scheduling algorithm for input-queued switches, *IEEE/ACM transactions on networking* **7(2)**, 188–201 (1999).

[30] Lee, Hyun Yeop and Hwang, Frank K and Carpinelli, John D, A new decomposition algorithm for rearrangeable Clos interconnection networks, *IEEE Transactions on Communications* **44(11)**, 1572–1578 (1996).

[31] Chang, Cheng-Shang and Chen, Wen-Jyh and Huang, Hsiang-Yi, On service guarantees for input-buffered crossbar switches: a capacity decomposition approach by Birkhoff and von Neumann, *Quality of Service, 1999. IWQoS'99. 1999 Seventh International Workshop on*, 79–86 (1999).

[32] Chang, Cheng-Shang and Chen, Wen-Jyh and Huang, Hsiang-Yi, Birkhoff-von Neumann input buffered crossbar switches, *IEEE INFOCOM. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings*. IEEE **3**, 1614–1623 (2000).

[33] Birkhoff, Garret, *Tres observaciones sobre el algebra lineal*, Univ. Nac. Tucumán Rev. Ser. A **5**, 147–151 (1946).

[34] Von Neumann, John, A certain zero-sum two-person game equivalent to the optimal assignment problem, *Contributions to the Theory of Games* **2**, 5–12 (1953).

[35] McKeown, Nick and Varaiya, Pravin and Walrand, Jean, Scheduling cells in an input-queued switch, *Electronics Letters* **29(25)**, 2174–2175 (1993).

[36] Chang, Cheng-Shang and Lee, Duan-Shin and Jou, Yi-Shean, Load balanced Birkhoff–von Neumann switches, part I: one-stage buffering, *Computer Communications* **25(6)**, 611–622 (2002).

[37] Dong, Ziqian and Rojas-Cessa, Roberto, *Non-blocking memory-memory-memory Clos-network packet switch*, IEEE, 34th IEEE Sarnoff Symposium **32**, 1–5 (2011).

[38] Dong, Ziqian and Rojas-Cessa, Roberto and Oki, Eiji, IEEE, Memory-memory-memory Clos-network packet switches with in-sequence service **21**, *High Performance Switching and Routing (HPSR), 2011 IEEE 12th International Conference on*, 121–125 (2011).

[39] Dong, Ziqian and Rojas-Cessa, Roberto and Oki, Eiji, Buffered clos-network packet switch with per-output flow queues, *Electronics letters* **47(1)**, 32–34 (2011).

Oladele Theophilus Sule received the B.Sc. degree in Physics from Benson Idahosa University, Benin City, Nigeria, in 2008. He received the M.Sc. and Ph.D. degrees in Electrical Engineering from New Jersey Institute of Technology, Newark, in 2012 and 2018, respectively. His research interests include large scale and high performance switches, next generation network architecture, and data center networking.

Roberto Rojas-Cessa (S'97–M'01–SM'11) is currently a Professor in the Department of Electrical and Computer Engineering at New Jersey Institute of Technology (NJIT), Newark, NJ. His research interest includes packet switching, network measurement and security, optical and emergency communications.