

On the Maximal Throughput of Networks with Finite Buffers and its Application to Buffered Crossbars

Paolo Giaccone Emilio Leonardi
Dipartimento di Elettronica
Politecnico di Torino, Italy

Devavrat Shah
CS/EE Department
Stanford University, USA

Abstract—The advent of packet networks has motivated many researchers to study the performance of networks of queues in the last decade or two. However, most of the previous work assumes the availability of infinite queue-size. Instead, in this paper, we study the maximal achievable throughput in a flow-controlled lossless network with finite-queue size. In such networks, throughput depends on the packet scheduling policy utilized. As the main of this paper, we obtain a dynamic scheduling policy that achieves the maximal throughput (equal to the maximal throughput in the presence of infinite queue-size) with a minimal finite queue-size at the internal nodes of the network. Though the performance of the policy is ideal, it is quite complex and hence difficult to implement. This leads us to a design of simpler and possibly implementable policy. We obtain a natural trade-off between throughput and queue-size for this policy.

We apply our results to the packet switches with buffered crossbar architecture. We propose a simple, implementable, distributed scheduling policy which provides high throughput in the presence of minimal internal buffer. We also obtain a natural trade-off between throughput, internal speedup and buffer-size providing a switch designer with a gamut of designs.

To the best of authors' knowledge, this is one of the first attempts to study the throughput for general networks with finite queue-size. We believe that our methods are general and can be useful in other contexts.

I. INTRODUCTION

Flow-controlled lossless network architectures (like ATM networks [12], [19] or wormhole routed networks [4], [17], [26]) have failed in the context of Internet. However, such network architectures are still widely prevalent in several other contexts such as storage area networks [34], interconnection networks for parallel computing [13], etc. In this paper, we study the throughput performance of such flow-controlled lossless networks.

The seminal work of Tassioulas and Ephremides [31] pioneered the research for studying the maximal throughput of controlled networks (also called constrained queueing systems in [31]) with infinite queue-size. For example, the methods of [31] have been utilized in the context of switching [1], [6], [15], [20], [33], satellite and wireless networks [24], [25], etc. Although these results are quite general, they assume the availability of infinite queue-size at all the nodes of the network. In actual applications, queue-sizes are always finite. This is a major limitation of the results of [31] as well as results known in the network theory in general.

In this paper, motivated to analyze performance of networks in the absence of the assumption of infinite queue-size, we study the maximal achievable throughput in flow-controlled lossless networks with finite queue-sizes at the internal nodes of the network. Only the ingress nodes have infinite-size queues to allow us to define the achievable throughput. Our work can be seen as an extension of the results of [31] in the sense that it gets rid of the assumption of infinite queue-size inside the network.

As an application of our results, we evaluate the maximal achievable throughput in the packet switch architecture built around a crossbar with buffered crosspoints. Such packet switches have been of a huge recent interest due to the recent advances in the technology [35]. We propose a novel distributed scheduling policy, called DMWF, which is shown to be stable under admissible i.i.d. Bernoulli traffic if either enough internal buffer is present and/or enough internal speedup is available at the crossbar. In particular, we evaluate the natural tradeoff between throughput, speedup and buffer-size.

We would like to note that though the contribution of this paper is mainly theoretical, we believe that the results obtained in this paper will be useful in the design of network, in sizing buffers and in the design of scheduling policies.

A. Organization

In Section II, we introduce the notation and the main assumptions of the paper. Then, we present the known results about the maximal throughput for the network of infinite queue-size in Section II-B. In Section III, we present our main results. Finally, in Section IV, we introduce the buffered crossbar switch architecture and apply our results to obtain distributed scheduling policy for buffered crossbar. The proofs of the theorems are presented in the appendix of the paper.

II. SYSTEM MODEL AND NOTATION

We consider a network of discrete-time physical queues (or stations), handling J customer flows. Customers belonging to flow j , $1 \leq j \leq J$, enter the network at a station, receive service at each station along an acyclic path and leave the network. The number of stations (hops) traversed by customers of flow j is h_j . We assume that the routing of each flow is deterministic. Figure 1 depicts an example of such network.

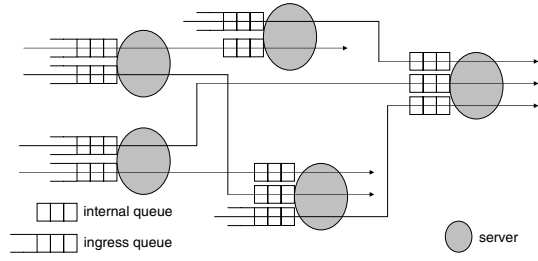


Fig. 1. Example of flow controlled network with 5 servers and 6 flows.

Customers belonging to the same flow, and stored at the same physical queue, form a *virtual queue*. The number of virtual queues in a network is denoted by Q . We denote with v_q the q -th virtual queue in the network. Let $q(j, h)$, with $h = 1, \dots, h_j$, be the index of the queue traversed by flow j at the h -th hop. Virtual queue $v_{q(j,1)}$ is also called “ingress queue” for flow j . The set of all the flow ingress virtual queues $\{v_{q(j,1)}, 1 \leq j \leq J\}$ is denoted with Φ_I , whose cardinality is J since each ingress queue is associated with a flow. The set of all the remaining virtual queues, called “internal queues”, is denoted with Φ_M . Note that Φ_M includes also “egress queues”, i.e. queues traversed by customers just before leaving the network. We now introduce some mapping functions. $f(q)$ maps queue index q to the index of its corresponding ingress queue (first queue): $f(q(j, h)) = q(j, 1)$; note that $f(q) = q$ if $q \in \Phi_I$. Function $u(q)$ returns the index of the queue upstream to queue v_q and it is defined for all the queues except for the ingress queues: $u(q(j, h)) = q(j, h - 1)$, $2 \leq h \leq h_j$. Function $p(q)$ returns the index of queue downstream to queue v_q and it is defined for every queue except the egress queues: $p(q(j, h)) = q(j, h + 1)$, $1 \leq h \leq h_j - 1$. Function $j(q)$ returns the index of the flow traversing queue v_q .

Servers are associated with physical queues. A server provides service to customers queued at the logical queues (virtual queues) located at its physical queue according to a service policy. Packets are assumed of fixed length normalized to a time slot. All the servers in the network are synchronized. They start service at the beginning of a time slot.

All vectors are, by default, row vectors. Let $X(n) = [x_q(n)]_{q=1}^Q = [x_1(n) \ x_2(n) \ \dots \ x_Q(n)]$, be the vector whose q -th component $x_q(n)$ represents the number of customers present in v_q (i.e., its queue length¹) at the beginning of time n , with $n \in \mathbf{N}_+$. For the sake of easier notation, let $x_{j,h}(n)$ equal to $x_{q(j,h)}(n)$. We suppose the size of all the ingress virtual queues (belonging to Φ_I) to be infinite, whereas we assume all the other virtual queues along the flow paths (belonging to Φ_M) to be of *finite size*. All the virtual queues traversed by flow j can store up to l_j customers. A flow control mechanism inhibits the transmission of packets toward queues which are full, thus preventing buffer overflows.

The evolution of the number of customers at v_q is described

¹Note that “length” denotes the time-variable queue-occupancy, whereas “size” denotes the fixed maximum allowed queue-occupancy.

by $x_q(n + 1) = x_q(n) + e_q(n) - d_q(n)$, where $e_q(n)$ represents the number of customers that entered v_q (and thus its corresponding physical queue) in time interval $(n, n + 1]$, and $d_q(n)$ represents the number of customers departed from v_q in time interval $(n, n + 1]$. $E(n) = [e_q(n)]_{q=1}^Q$ is the vector of entrances in virtual queues, and $D(n) = [d_q(n)]_{q=1}^Q$ is the vector of departures from virtual queues; let $d_{j,h}(n)$ be defined equal to $d_{q(j,h)}(n)$. Given the network state, $X(n)$, the following *service constraints* on $D(n)$ express the fact that no services can be provided to empty queues and no queue overflows are allowed (thanks to the flow control mechanism):

$$D(n) \leq X(n) \quad \text{and} \quad D(n)\mathbf{R} \leq L - X(n) \quad (1)$$

being $L = [l_q]_{q=1}^Q$ the vector with all the queue sizes (assuming $l_q = \infty$ for $q \in \Phi_I$). This is equivalent to say: if queue q is empty ($x_q(n) = 0$) or the downstream queue to q is full ($x_{p(q)}(n) = l_{j(q)}$), then no service can be provided to queue q ($d_q(n) = 0$).

With this notation, the system evolution equation can be written as:

$$X(n + 1) = X(n) + E(n) - D(n) \quad (2)$$

The entrance vector $E(n)$ is sum of two terms: vector $A(n) = [a_q(n)]_{q=1}^Q$, representing the customers arrived at the system from outside the network of queues (we also call them external arrivals), and vector of recirculating customers, who are advancing along their paths inside the network, in time interval $(n, n + 1]$.

Since we assume deterministic routing, let the $Q \times Q$ matrix \mathbf{R} be the *routing matrix*, with binary elements $\mathbf{R}_{q_1 q_2} = 1$ iff $p(q_1) = q_2$ and $\mathbf{R}_{q_1 q_2} = 0$ otherwise. The evolution of virtual queues can be rewritten as:

$$X(n + 1) = X(n) + A(n) - D(n)(\mathbf{I} - \mathbf{R}) \quad (3)$$

where \mathbf{I} denotes the identity matrix.

We suppose that the external arrival process $\{A(n) : n \in \mathbf{N}_+\}$ is a stationary memoryless process, i.e. $A(n)$ are i.i.d. random vectors with average $E(A(n)) = \Lambda = [\lambda_q]_{q=1}^Q$. Note that, since external arrivals are directed only to ingress queues, $\lambda_q = 0$ if $q \in \Phi_M$.

At time slot n the scheduling policy selects the service vector $S(n)$, whose element $s_q(n)$ represents the amount of work provided to v_q during time slot n ; let $s_{j,h}(n) = s_{q(j,h)}(n)$. The departure vector $D(n)$ is related to $S(n)$ according to the following equation:

$$\sum_{t=0}^n D(t) = \left[\sum_{t=0}^n S(t) \right] \Rightarrow D(n) = \left[\sum_{t=0}^n S(t) \right] - \sum_{t=0}^{n-1} D(t)$$

In other words, the number of packets served from a queue is given by the amount (approximated to the integer part) of cumulative service provided to the queue. We define the difference between the two quantities by:

$$\Delta(n) = \sum_{t=0}^n S(t) - \sum_{t=0}^n D(t)$$

whose q -th element $\delta_q(n) \in [0, 1)$ represents the amount of work provided by the scheduling policy to the head-of-the-line packets of v_q at the end of time slot n . In this paper we restrict our investigation to the class of *dynamic* scheduling policies, i.e. those scheduling policies which select $S(n)$ on the only basis of the instantaneous network state information without requiring any knowledge on the traffic pattern.

In general we assume that the set of possible service vectors $S(n)$ is constrained by a system of linear equations representing the topological interference among services at virtual queues (*blocking constraints*):

$$S(n)\mathbf{K} \leq T \quad (4)$$

Matrix \mathbf{K} and vector T describe the blocking constraints in the services. For example, simple topological constraints are those expressing the fact that the sum of services provided in each time slot to all the virtual queues residing at the same physical queue is limited by the server capacity. However we do not exclude additional constraints which relate the behavior of virtual queues residing at different stations. Let $\widehat{\mathcal{D}}$ the set of non-negative $S(n)$ which satisfy the blocking constraints. We notice that $\widehat{\mathcal{D}}$ defines a polyhedral convex region. Let \mathcal{D} the set of all vertices of $\widehat{\mathcal{D}}$. For simplicity of notation, we assume that vectors in \mathcal{D} are integer valued. In order to avoid trivial cases, we assume that all vectors $\gamma^{(q)}$, with $1 \leq q \leq Q$, whose elements are all null except the q -th, which is unitary, belongs to $\widehat{\mathcal{D}}$, i.e. $\gamma^{(q)} = [0, 0, 0, \dots, 1, 0, 0, 0] \in \widehat{\mathcal{D}}$; in other words, every virtual queue in the network can potentially get service without violating the topological constraints. We remind that $S(n)$ must be chosen in such a way that the service constraints defined for $D(n)$ in (1) are not violated.

If the scheduling policy is *atomic*, i.e. packets are transmitted by servers in an ‘atomic’ fashion, without interleaving their transmission with packets residing in other virtual queues, then $S(n)$ is integer valued, and $D(n) = S(n)$ for any n . In this case, $X(n)$ is a DTMC (Discrete Time Markov Chain). In the more general case, $(X(n), \Delta(n))$ is a discrete time Markov process defined on a general state space [21]. In the latter case, let us define the workload vector:

$$Y(n) = X(n) - \Delta(n)(\mathbf{I} - \mathbf{R})$$

We notice that $(X(n), \Delta(n)) \Rightarrow Y(n)$ is a one to one correspondence. Furthermore, it is easy to verify that $Y(n)$ satisfies the following system evolution equation, derived by (3):

$$Y(n+1) = Y(n) + A(n) - S(n)(\mathbf{I} - \mathbf{R}) \quad (5)$$

Note that if the scheduling policy is atomic, then $X(n) = Y(n)$ for any n and (5) coincides with (3).

Finally, let us introduce the following useful positive convex functional:

Definition 1: Given a vector $Z \in \mathbb{R}_+^Q$, $Z = (z^{(k)}, 1 \leq k \leq Q)$, the positive convex functional $\|Z\|$ is defined as:

$$\|Z\| = \inf \left\{ \alpha \in \mathbb{R}_+ : \frac{1}{\alpha} Z \in \widehat{\mathcal{D}} \right\}$$

We notice that in the remainder of this paper we will refer to it with the improper term of ‘norm’; furthermore, the previous functional is coincident with the well-known Minkowski convex functional associated to $\widehat{\mathcal{D}}$. Under the $\|Z\|$ definition immediately follows that, for any $S(n) \in \widehat{\mathcal{D}}$, then $\|S(n)\| \leq 1$. If the policy is atomic, it chooses $S(n) = D(n) \in \mathcal{D}$. To better understand the meaning of such norm, we evaluate now $\|\mathbb{I}\|$, where \mathbb{I} is the vector with unitary elements, in a simple case. Consider a set of independent servers, each of them able to provide one unit of service per time slot, i.e. $\gamma^{(q)}$, $1 \leq q \leq Q$, is vertex of $\widehat{\mathcal{D}}$. If at most i virtual queues are located at each server, then $\|\mathbb{I}\| = i$. Indeed, the service vector $S = \mathbb{I}/i$ belongs to the boundary of $\widehat{\mathcal{D}}$.

A. Traffic and System Stability Definitions

Definition 2: A stationary traffic pattern is admissible if $\|\Lambda(\mathbf{I} - \mathbf{R})^{-1}\| < 1$.

Let $\rho = \|\Lambda(\mathbf{I} - \mathbf{R})^{-1}\|$. For the simplest case in which virtual queues residing at different servers are not topologically interacting, traffic is admissible iff no servers are overloaded; in addition, ρ represents the load of the heaviest loaded server in the network.

Definition 3: The system of queues is *stable* if:

$$\limsup_{n \rightarrow \infty} E(\|X(n)\|) < \infty$$

or equivalently:

$$\limsup_{n \rightarrow \infty} E(\|Y(n)\|) < \infty$$

i.e., the system is positive recurrent.

Note that the admissibility of traffic pattern is a necessary condition for the system of queue to be stable as shown in [31].

We say that the system is stable at point Λ if it is stable under every stationary memoryless external arrival processes $A(n)$ with average Λ .

Definition 4: We define as *stability region* the set of points Λ in correspondence of which the system of queues is stable.

Definition 5: We say that a system of queues is 1-efficient (or equivalently achieves 100% throughput), if it is stable under any admissible traffic pattern.

Definition 6: For any $0 < \rho < 1$, we say that the system of queues is ρ -efficient if it is stable under any traffic pattern such that $\|\Lambda(\mathbf{I} - \mathbf{R})^{-1}\| \leq \rho$.

B. Previous work

The problem of the definition of the stability region in complex systems of interacting queues under dynamic scheduling policies, has attracted significant attention in the last decade from the research community since the pioneering work [31].

In [31], applying the Lyapunov function methodology, it has been shown that a system of interacting queues whose size is infinite achieve 100% throughput, if atomic max-scalar scheduling policy \mathcal{P}_{MS} is applied at each node of the network. According to \mathcal{P}_{MS} , at each time slot n the departure vector is selected as follows:

$$D(n) = S(n) = \arg \max_{Z \in \mathcal{D}} Z(\mathbf{I} - \mathbf{R})X(n)^T \quad (6)$$

The result in [31] has been generalized and adapted to different application contexts in the last years. As matter of example we just briefly recall some of the related works.

In the switching context, several studies have been aimed at the definition of the stability region in Input-Queued (IQ) switching architectures built around a bufferless crossbar: papers [1], [15], [20], [30], [33] have proposed different extensions of \mathcal{P}_{MS} , which have been shown to be 1-efficient; stability properties for simpler scheduling policies have been also studied in [6], [14]; in [2], [3], [15], finally, the problem of the definition of the stability region in networks of IQ switches has been considered.

In the context of the satellite and wireless networks, generalizations of \mathcal{P}_{MS} have been recently proposed and shown to be 1-efficient in [24], [25], [32].

All the previous works, however, have considered system of infinite-size queues. In this paper, for the first time, to the best of our knowledge, we extend the investigation about the stability region in systems and networks of queues of finite size subject to some form of flow control which prevents packet losses.

C. Stability criteria

The stochastic Lyapunov function is a powerful tool to prove stability (i.e., positive recurrency) of Markovian systems. In this subsection we briefly report one of the main results related to the Lyapunov function methodology, which will be used in the remainder of this paper; we refer the interested reader to [11], and [21], [25] for more details on the extension to general state space Markovian processes.

Theorem 1: Let $Z(n)$ be an irreducible Q -dimensional Markov chain (or, general space Markov process), whose elements $z_l(n), l = 1, 2, \dots, Q$ are non-negative, i.e., $Z(n) \in \mathbb{N}_+^Q$ (or, $Z(n) \in \mathbb{R}_+^Q$). If there exists a non-negative valued function $\{\mathcal{L} : \mathbb{R}_+^Q \rightarrow \mathbb{R}_+\}$ such that:

$$E[\mathcal{L}(Z(n+1)) - \mathcal{L}(Z(n)) \mid Z(n)] < \infty \quad (7)$$

and

$$\limsup_{\|Z(n)\| \rightarrow \infty} \frac{E[\mathcal{L}(Z(n+1)) - \mathcal{L}(Z(n)) \mid Z(n)]}{\|Z(n)\|} < -\epsilon \quad (8)$$

for some $\epsilon > 0$, then $Z(n)$ is positive recurrent, and

$$\limsup_{n \rightarrow \infty} E[\|Z(n)\|] < \infty$$

Inequality (7) requires that the increments of the Lyapunov function $\mathcal{L}(Z)$ are finite on average. The second inequality (8) requires that, for large values of $\|Z\|$, the average increment in the Lyapunov function from time n to time $n+1$ is negative. The intuition behind this result is that the system must be such that a negative feedback exists, which is able to pull the system toward the empty state, thus making it ergodic. For these reasons, inequality (8) is often referred as the Lyapunov function drift condition.

In our case, $Z(n)$ represents the number of packets in the network of queue $X(n)$ or the workload $Y(n)$, whose evolution is given by (3) or (5). It is immediate to verify that

constraint (7) can be always met when all the moments of $A(n)$ are finite; in particular, restricting to quadratic Lyapunov functions, it is sufficient that the second moment of $A(n)$ is finite.

III. PERFORMANCE OF NETWORK OF FINITE QUEUES

Here we present our main results. In Section III-A we show that 100% throughput can be obtained in any network of finite, flow controlled interacting queues, for $l_j \geq 1$. To this end, we define the optimal dynamic scheduling policy \mathcal{P}_1 . Since policy \mathcal{P}_1 (i) is not atomic, i.e. servers provide fractional services to packets stored at head of the virtual queues, (ii) requires the servers to coordinate their decisions at each time slot, then its implementability results problematic in several application contexts.

In Section III-B we propose the atomic dynamic scheduling policy \mathcal{P}_2 whose complexity is similar to \mathcal{P}_{MS} defined for infinite queue networks. \mathcal{P}_2 , similarly to \mathcal{P}_{MS} , requires a continuous exchange of state information among network servers, but it can allow servers to take local decisions in an uncoordinated fashion, when considering simple network configurations, thus resulting significantly less complex than \mathcal{P}_1 . We show that \mathcal{P}_2 is ρ -efficient when enough buffer inside the network is provided, thus estimating the trade-off between network buffers and achievable throughput.

A. Optimal policy

1) **Policy definition:** Consider the following policy, called \mathcal{P}_1 , in vectorial format:

$$S(n) = \arg \max_{Z \in \mathcal{D}} Z(\mathbf{I} - \mathbf{R})\mathbf{M}(n)(2Y(n) - Z(\mathbf{I} - \mathbf{R}))^T \quad (9)$$

where $\mathbf{M}(n)$ is a $Q \times Q$ matrix, non-null only on its diagonal where, for $q = 1, \dots, Q$:

$$\mathbf{M}_{qq}(n) = \begin{cases} 1 & \text{if } q \in \Phi_I \\ \frac{y_{f(q)}(n)}{l_{j(q)}} & \text{if } q \in \Phi_M \end{cases}$$

We now express the policy in scalar format (for the sake of easier notation, we omit (n) when not necessary). Observe that:

$$[S(\mathbf{I} - \mathbf{R})]_q = \begin{cases} s_q & \text{if } q \in \Phi_I \\ s_q - s_{u(q)} & \text{if } q \in \Phi_M \end{cases}$$

and multiplying by \mathbf{M} :

$$[S(\mathbf{I} - \mathbf{R})\mathbf{M}]_q = \begin{cases} s_q & \text{if } q \in \Phi_I \\ (s_q - s_{u(q)}) \frac{y_{f(q)}}{l_{j(q)}} & \text{if } q \in \Phi_M \end{cases}$$

Hence, if we define that $y_{p(q)}/l_{j(q)} = 0$ if queue q is an egress

queue, the first adder in (9) becomes:

$$\begin{aligned}
f_1(S) &= S(\mathbf{I} - \mathbf{R})\mathbf{M}Y^T = \\
&\sum_{q \in \Phi_I} s_q y_q + \sum_{q \in \Phi_M} (s_q - s_{u(q)}) \frac{y_{f(q)}}{l_{j(q)}} = \\
&\sum_{q \in \Phi_I} s_q y_q \left(1 - \frac{y_{p(q)}}{l_{j(q)}}\right) + \sum_{q \in \Phi_M} s_q y_{f(q)} \left(\frac{y_q - y_{p(q)}}{l_{j(q)}}\right) = \\
&\sum_{j=1}^J \frac{y_{j,1}}{l_j} \left[s_{j,1} (l_j - y_{j,2}) + \sum_{h=2}^{h_j} s_{j,h} (y_{j,h} - y_{j,h+1}) \right] \quad (10)
\end{aligned}$$

whereas the second adder in (9):

$$\begin{aligned}
f_2(S) &= S(\mathbf{I} - \mathbf{R})\mathbf{M}[S(\mathbf{I} - \mathbf{R})]^T = \\
&\sum_{q \in \Phi_I} s_q^2 + \sum_{q \in \Phi_M} (s_q - s_{u(q)})^2 \frac{y_{f(q)}}{l_{j(q)}} = \\
&\sum_{j=1}^J s_{j,1}^2 + \sum_{j=1}^J \frac{y_{j,1}}{l_j} \sum_{h=2}^{h_j} (s_{j,h}^2 + s_{j,h-1}^2 - 2s_{j,h}s_{j,h-1}) = \\
&\sum_{j=1}^J s_{j,1}^2 + \sum_{j=1}^J \frac{y_{j,1}}{l_j} \sum_{h=2}^{h_j} (s_{j,h}^2 + s_{j,h-1}^2 - 2s_{j,h}s_{j,h-1}) = \\
&\sum_{j=1}^J s_{j,1}^2 + \sum_{j=1}^J \frac{y_{j,1}}{l_j} \left(s_{j,1}^2 + 2 \sum_{h=2}^{h_j-1} s_{j,h}^2 + s_{j,h_j}^2 - \right. \\
&\quad \left. 2 \sum_{h=1}^{h_j-1} s_{j,h}s_{j,h+1} \right) \quad (11)
\end{aligned}$$

By combining (10) and (11), policy \mathcal{P}_1 becomes:

$$S = \arg \max_{Z \in \hat{\mathcal{D}}} f(Z) \quad (12)$$

with $f(Z) = 2f_1(Z) - f_2(Z)$.

Observe that according to policy \mathcal{P}_1 , by construction, service is never provided to empty virtual queues, thus satisfying one of the service constraints. This can be easily seen by observing that \mathcal{P}_1 can be equivalently defined as:

$$\begin{aligned}
S(n) &= \arg \min_{Z \in \hat{\mathcal{D}}} \left\{ \right. \\
&\left. [Y(n) - Z(n)(\mathbf{I} - \mathbf{R})]\mathbf{M}(n)[Y(n) - Z(n)(\mathbf{I} - \mathbf{R})]^T \right\}
\end{aligned}$$

and observing that the minimum is always achieved when all the elements of $[Y(n) - Z(n)(\mathbf{I} - \mathbf{R})]$ are non negative.

The second service constraint, which avoids buffer overflows, is not always precisely met by \mathcal{P}_1 . However it is easy to realize that according to \mathcal{P}_1 , $Y(n) \leq L + c\mathbb{I}$, for all n , being c the maximum amount of service that any server in the network can provide in a time slot. As a conclusion, to avoid buffer overflow is sufficient to provide any virtual queue with an extra amount of memory (called slack-buffer) equal to c .

2) **Policy performance:** Now we state our main theorem, whose proof is reported in Appendix I.

Theorem 2: Under admissible Bernoulli traffic, policy \mathcal{P}_1 achieves 100% throughput when the buffer size l_j (not counting the slack buffer) of any internal queue q traversed by flow j satisfies the following relation:

$$l_j \geq 1 \quad \text{for } j = 1, \dots, J$$

3) **Implementation issue:** Since \mathcal{P}_1 is not atomic, it selects the best service vector S in the set $\hat{\mathcal{D}}$ and this does not guarantee that S is a integer departure vector; in simpler words, $s_q \in [0, 1]$. As a consequence, the direct implementation of policy \mathcal{P}_1 requires servers to provide fractional services to packets stored at head of the virtual queues according to a weighted processor sharing policy.

Moreover, according to policy \mathcal{P}_1 , packets are transferred through queues in a ‘‘cut-through’’ fashion, since servers may start the transmission of non completely received packets. We notice that non-atomic scheduling policies exploiting ‘‘cut-through’’ switching have been proposed and implemented in the contexts of wormhole networks [7], [10], [26].

At last, \mathcal{P}_1 must be implemented in a centralized fashion by a scheduler which has the complete view of the queues state of the network. The high implementation complexity of this policy has motivated our investigation on the performance of the following policy.

B. Low complexity policy

1) **Policy definition:** Consider the following policy, called \mathcal{P}_2 :

$$S = \arg \max_{Z \in \hat{\mathcal{D}}} Z(\mathbf{I} - \mathbf{R})\mathbf{M}Y^T$$

In other words, policy \mathcal{P}_2 maximizes the scalar product of the service vector Z and the weight vector $W = (\mathbf{I} - \mathbf{R})\mathbf{M}Y^T$. Due to the linearity of the scalar product, \mathcal{P}_2 guarantees the vector S to be a vertex of $\hat{\mathcal{D}}$, i.e. $S(n) \in \mathcal{D}$. Hence, \mathcal{P}_2 is an atomic policy and $X(n) = Y(n)$. Formally, we can say that \mathcal{P}_2 can be expressed as:

$$D = \arg \max_{Z \in \mathcal{D}} Z(\mathbf{I} - \mathbf{R})\mathbf{M}X^T$$

Following the same reasoning to obtain (10), a generic queue q is associated with the following weight w_q :

$$w_q = \begin{cases} x_q \left(1 - \frac{x_{p(q)}}{l_{j(q)}}\right) & \text{if } q \in \Phi_I \\ x_{f(q)} \left(\frac{x_q - x_{p(q)}}{l_{j(q)}}\right) & \text{if } q \in \Phi_M \end{cases} \quad (13)$$

then policy \mathcal{P}_2 can be rewritten as:

$$D = \arg \max_{Z \in \mathcal{D}} \sum_{q=1}^Q z_q w_q \quad (14)$$

We define the policy such that $d_q = 0$ when $w_q = 0$; note also that $d_q = 0$ when $w_q < 0$. Hence, policy \mathcal{P}_2 satisfies the service constraints: if $x_q = 0$ or $x_{p(q)} = l_{j(q)}$ then $w_q \leq 0$ and then $d_q = 0$ as expected.

2) **Policy performance:** We claim the main result about \mathcal{P}_2 , whose proof is reported in Appendix II.

Theorem 3: Under admissible Bernoulli traffic, policy \mathcal{P}_2 is ρ -efficient when the buffer size l_j of any internal queue q traversed by flow j , with h_j hops, satisfies the following relation:

$$l_j > \frac{(h_j - 1)\|\mathbb{I}\|}{2(1 - \rho)} \quad \text{for } j = 1, \dots, J$$

recalling that $\rho = \|\Lambda(\mathbf{I} - \mathbf{R})^{-1}\|$, and \mathbb{I} is the vector with unitary elements.

A special case applies for networks with at most two hops, like the switches built around buffered crossbars and discussed in Section IV. We can claim the following:

Corollary 1: Under admissible Bernoulli traffic, for a network with $h_j \leq 2$ for all j , policy \mathcal{P}_2 achieves 100% throughput, when $\rho < 0.5$ for any $l_j \geq 1$, being ρ the maximum offered load for a single queue in the network.

The proof is reported in Appendix III. From this corollary, it results that any choice of l_j the network is 0.5-efficient, under the condition that no packet routes are longer than two hops.

3) **Implementation issue:** Policy \mathcal{P}_2 is an atomic policy equivalent to \mathcal{P}_{MS} of (6), but with different weights assigned to the internal queues. Indeed, \mathcal{P}_2 and \mathcal{P}_{MS} solve the same optimization problem since they both share the same linear structure of the cost function and the same space \mathcal{D} of feasible departure vectors.

Both policies require a continuous exchange of information between neighbor servers, but in addition \mathcal{P}_2 requires locally at each server the information about the length of the ingress queue of the corresponding flows. Note that this length should be propagated downstream from the ingress queue to all the internal queues, along the flow path: this fact can be exploited to ease the implementation.

In general, given the state of all the queues, \mathcal{P}_2 is executed by a central scheduler, as also observed by [31]. However, in particular (but also interesting) cases, the policy can be computed in a distributed fashion, locally on each set of queues and servers which are coupled by the blocking constraints. This fact is indeed exploited in the following section to devise a computationally efficient scheduling policy for packet switches.

IV. APPLICATION TO PACKET SWITCHES BASED ON BUFFERED CROSSBARS

Recently, switches built around crosspoint buffered crossbars have been shown to be very promising solutions for the design of fast and scalable switching architectures. A basic model for a switch with internal buffered crossbar is depicted in Fig. 2. To avoid the negative effects of the head-of-the-line blocking phenomenon, inputs cards adopts Virtual Output Queue (VOQ) scheme, according to which packets are stored at inputs in per-destination virtual queues.

Each crosspoint of the crossbar is provided with an internal buffer of size L : internal buffers are in one-to-one correspondence with input VOQs. We refer to this architecture as

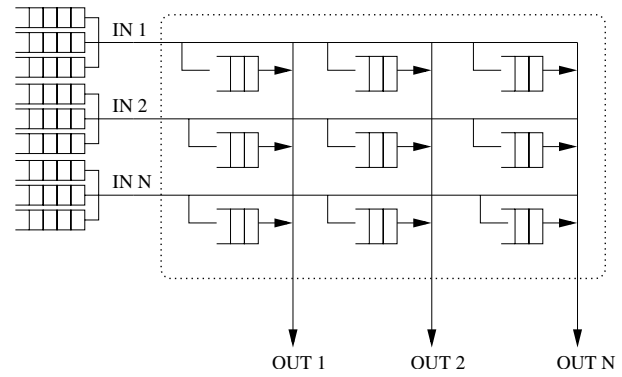


Fig. 2. The $N \times N$ CICQ architecture with VOQ and buffered crosspoints

Combined Input and Crossbar Queued (CICQ) switch. A flow control mechanism from each crosspoint to the corresponding VOQ avoids to overflow the internal buffer.

Assume time to be slotted, and packets to be of fixed size. With respect to pure input queued switches, the scheduling policies in CICQ switches can be simpler. The scheduling decision, indeed, can be taken in a local uncoordinated fashion by an arbiter at each input, selecting a non-full internal buffer to which transferring a packet, and by an arbiter at each output, selecting an internal buffer from which transferring a packet. We refer in the following to this class of schedulers as “uncoordinated schedulers”.

Uncoordinated schedulers can be efficiently distributed, parallelized, and pipelined. Mainly for this reason, CICQ switches are widely considered scalable and timely. Note that, in uncoordinated schedulers, we admit that inputs and outputs can exchange some information about the state of the queues, but we assume the scheduling decision to be local. Furthermore, uncoordinated schedulers cannot be implemented in pure IQ switches, since coordination is required at inputs to avoid multiple transmissions toward the same output.

Here, we restrict our discussion to uncoordinated schedulers.

An overview of the evolution of CICQ switches has been recently proposed in [35], but also [28] provides a wide introduction to CICQ switches. We refer to both cited papers for the main algorithms proposed so far to control the CICQ architecture, aimed at providing high throughput, or supporting QoS or variable size packets [9]. The two main families of input arbiters and output arbiters proposed and studied so far have been the following:

- round-robin based: the queue is selected according to a round robin (RR) mechanism [27], [28], [29], or to a weighted round robin (WRR) [5], or to a weighted fair queueing scheme (WFQ) [5];
- queue-state based: the queue with largest length (LQF) [8], or the largest waiting time of the HoL cell (OCF) [8], [23], or the largest/shortest internal queue length [22], is selected.

Note that the input arbiters can select a VOQ queue among the

VOQs which are not inhibited by the flow control mechanism.

Unfortunately, so far general theoretical results have been missing about the stability of CICQ for speedup $S_P < 2$. Many papers have addressed the case $S_P = 1$ but proving stability properties only under some ideal traffic scenarios, for example when the arrival rates for each input output port are known (e.g., in the case of uniform traffic). The simplest scheme of CICQ is based on RR-RR (notation is: “input arbiter”- “output arbiter”) [27] and $L = 1$; this scheme has been proved to be stable only for uniform traffic, indeed it has been shown to be unstable when the traffic is non-uniform ([8] and [35]). For the RR-RR scheme it has been shown [28] by simulation that small buffers ($L = \Theta(N)$) cannot be sufficient to provide 100% throughput, unless some moderate speedup is introduced. When adopting LQF-RR and $L = 1$, the CICQ has been proved to achieve the 100% throughput just when input/output pair loading is $\leq 1/N$ [8]. Many other variants have been proposed (like in [5], [22], [29], [35]), achieving high throughput under non-uniform scenarios, but their performance has been shown only by simulation.

How to dimension L has been discussed by many papers, which have related L only to the round trip time delay d_{RTT} of the flow control mechanism from the internal crosspoint to the input arbiters. In order to sustain a line rate r , L should be set larger coarsely than the product $d_{RTT} \times r$.

For $S_P = 2$, perfect emulation of an output queued switch (both FIFO and non-FIFO) [18] can be performed. In other words, $S_P = 2$ is sufficient to achieve 100% throughput, work-conservation and perfect delay control under any admissible traffic. The requirement for L is minimal, since it is aimed just at compensating for d_{RTT} .

To our best knowledge, no theoretical results are known which prove that CICQs with $S_P < 2$, exploiting uncoordinated schedulers, can achieve 100% throughput under any admissible traffic pattern.

Now observe that a CICQ switch can be modeled as a flow controlled network, with one server for each input (corresponding to the input arbiter) and with one server for each output (corresponding to the output arbiter). The flow control is from the internal buffers to the corresponding input arbiters. Hence, we can particularize to this context the general results obtained in the previous section. We restrict our investigation to policy \mathcal{P}_2 which can be easily implemented in a CICQ as an uncoordinated scheduler.

A. Scheduling algorithms for CICQ

Let x_{ij} be the length of VOQ from input i to output² j . Let b_{ij} be the length of the corresponding internal buffer; $0 \leq b_{ij} \leq L$, and when $b_{ij} = L$ the flow control mechanism inhibits the services from the corresponding VOQ: we assume that the flow control is immediate. Departure vector D comprises the services provided by the input arbiters and the output arbiters: d_{ij}^I describes the departure from the VOQ

²With abuse of notation, here j stands either for a flow identifier or an output.

corresponding to x_{ij} , whereas d_{ij}^O describes the departure from the internal buffer corresponding to b_{ij} . The set \mathcal{D} of all possible departing vectors is given by all D such that

$$\sum_{j=1}^N d_{ij}^I \leq 1 \quad \forall i \quad \text{and} \quad \sum_{i=1}^N d_{ij}^O \leq 1 \quad \forall j$$

which describe the blocking constraints of (4) in the context of a CICQ switch.

We particularize the policy \mathcal{P}_2 by showing that it can be easily implemented in an uncoordinated fashion. Indeed, revisiting (14), \mathcal{P}_2 selects the departing vector according to:

$$D = \arg \max_{D \in \mathcal{D}} \sum_{j=1}^N \frac{x_{j,1}}{L} (d_{j,1}(L - x_{j,2}) + d_{j,2}x_{j,2}) = \arg \max_{D \in \mathcal{D}} \sum_{i=1}^N \sum_{j=1}^N x_{ij} (d_{ij}^I(L - b_{ij}) + d_{ij}^O b_{ij}) \quad (15)$$

Let MWF be the “maximum weight first” policy, which serves the queue with the maximum *strictly* positive weight. D satisfying (15) can be obtained by:

- maximizing, for each input i , the product $d_{ij}^I \times x_{ij}(L - b_{ij})$ among all possible j ; this is MWF policy;
- maximizing, for each output j , the product $d_{ij}^O \times x_{ij}b_{ij}$ among all possible i ; this is again MWF policy.

As a consequence policy \mathcal{P}_2 operating a CICQs can be redefined as DMWF (Dual Maximum Weight First) according to the following algorithmic description:

- at each time slot, associate to each VOQ a weight $w_{ij}^I = x_{ij}(L - b_{ij})$;
- select at each input the non inhibited VOQ which maximizes w_{ij}^I over all $j = 1, \dots, N$;
- at each time slot, associate to each internal buffer a weight $w_{ij}^O = x_{ij}b_{ij}$;
- select at each output the non-empty internal buffer which maximizes w_{ij}^O over all $i = 1, \dots, N$.

Thanks to corollary 1, DMWF is ρ -efficient if $\rho < 0.5$ for any $L \geq 1$. Note that in the case $L = 1$, then DMWF degenerates into LQF-LQF scheduler.

If we now apply Theorem 3, in a CICQ switch $\|\mathbb{I}\| = N$ since N are the queues conflicting in the same input/output arbiter. Hence, in general L should be set such that $L > N/(1 - \rho)/2$. To summarize, we can claim the following:

Corollary 2: Under admissible admissible Bernoulli traffic, in a CICQ switch policy DMWF is ρ -efficient for $L \geq L_{\min}$ with

$$L_{\min} = \begin{cases} 1 & \text{if } \rho < 0.5 \\ \left\lceil \frac{N}{2(1 - \rho)} \right\rceil & \text{if } 0.5 \leq \rho < 1 \end{cases}$$

where ρ is the maximum offered load to an input and output port of the switch.

The result of corollary 2 can be restated also as follows: the sustainable load is at least:

$$\rho = \begin{cases} 0.5 & \text{for } 1 \leq L \leq N \\ 1 - \frac{N}{2L} & \text{for } L > N \end{cases}$$

or equivalently:

Corollary 3: Under admissible Bernoulli traffic, the minimum speedup to guarantee 100% throughput in a CICQ switch adopting DMWF policy, is

$$S_P = \begin{cases} 2 & \text{for } 1 \leq L \leq N \\ \frac{2L}{2L - N} & \text{for } L > N \end{cases}$$

This proves the existence of a tradeoff between throughput (or speedup needed) and L under DMWF.

V. CONCLUSIONS

We have considered a network/system of interacting queues with internal queues of finite size. A flow control mechanism from each queue prevents losses to occur.

We devised two stable policies, \mathcal{P}_1 and \mathcal{P}_2 , the first achieving 100% throughput and the second ρ -efficient, under Bernoulli i.i.d. traffic. Policy \mathcal{P}_1 requires to solve a quadratic-form optimization problem on the state of workload given to each queue. This policy can be very complex to implement, but requires a minimal amount of one buffer location for each queue. On the contrary, policy \mathcal{P}_2 is based on the solution of a linear-form optimization problem on the state of occupation of the queues. This policy is very similar to the max-scalar policy proposed in the past for interacting queues with infinite-size buffer, and its computational complexity is lower than \mathcal{P}_1 . But the requirement on the amount of buffer is larger than \mathcal{P}_2 .

As example of application of our general theoretical results, we have considered a $N \times N$ input queued switch built around a buffered crossbar. In this case, \mathcal{P}_2 degenerates into an uncoordinated scheduling policy, called DMWF, in which each input and output arbiter can choose among N queues on the basis of the highest weight assigned to each queue. We have discussed the minimum buffer requirement for the internal queues, and its tradeoff with the allowed speedup and throughput in the crossbar.

REFERENCES

- [1] M. Ajmone Marsan, A. Bianco, P. Giaccone, E. Leonardi, F. Neri, "Packet-Mode Scheduling in Input-Queued Cell-Based Switch", *IEEE/ACM Transactions on Networking*, vol. 10, n. 5, Oct. 2002, pp. 666-678
- [2] M. Ajmone Marsan, P. Giaccone, E. Leonardi, F. Neri, "On the stability of local scheduling policies in networks of packet switches with input queues", *IEEE Journal on Selected Areas in Communications*, vol. 21, n. 4, May 2003, pp. 642-655
- [3] M. Andrews, L. Zhang, "Achieving Stability in Networks of Input-Queued Switches", *IEEE INFOCOM 2001*, Anchorage, Alaska, USA, Apr. 2001, pp. 1673-1679
- [4] N.J. Boden *et al.*, "Myrinet: a gigabit-per-second local area network", *IEEE Micro*, vol. 15, n. 1, Feb. 1995, pp. 29-36
- [5] N. Chrysos, M. Katevenis, "Weighted Fairness in Buffered Crossbar Scheduling", *IEEE HPSR 2003*, Torino, Italy, June 2003, pp. 17-22
- [6] J.G. Dai, B. Prabhakar, "The throughput of data switches with and without speedup", *IEEE INFOCOM 2000*, Tel Aviv, Israel, Mar. 2000, pp. 556-564
- [7] J. Duato, "A necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks", *IEEE Trans. on Parallel and Distributed Systems*, vol. 6, n. 10, Oct. 1995, pp. 1055-1067
- [8] T. Javadi, R. Magill, T. Hrabik, "A high throughput scheduling algorithm for a buffered crossbar switch fabric", *IEEE ICC 2001*, June 2001, pp. 1581-1591
- [9] M. Katevenis, G. Passas, D. Simos, I. Papaefstathiou, N. Chrysos, "Variable Packet Size Buffered Crossbar (CICQ) Switches", *IEEE ICC 2004*, Paris, France, June 2004
- [10] P. Kermani, L. Kleinrock, "Virtual Cut-through: A New Computer Communication Switching Technique", *Computer Networks*, vol. 3, n. 3, Sep. 1979, pp. 267-286
- [11] P.R. Kumar, S.P. Meyn, "Stability of Queuing Networks and Scheduling Policies", *IEEE Trans. on Automatic Control*, vol. 40, n. 2, Feb. 1995, pp. 251-260
- [12] H.T. Kung, K. Chang "Receiver-oriented adaptive buffer allocation in credit-based flow control for ATM networks" *IEEE INFOCOM '95*, Boston, MA, USA, Apr. 1995, pp. 239-252
- [13] F.T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees and Hypercubes*, Morgan Kaufmann, 1991
- [14] E. Leonardi, M. Mellia, F. Neri, M. Ajmone Marsan, "On the Stability of Input-Queued Switches with Speedup", *IEEE/ACM Trans. on Networking*, vol. 9, n. 1, Feb. 2001, pp. 104-118
- [15] E. Leonardi, M. Mellia, M. Ajmone Marsan, F. Neri, "On the Throughput Achievable by Isolated and Interconnected Input-Queuing Switches under Multiclass Traffic", *IEEE INFOCOM 2002*, New York, NY, USA, June 2002
- [16] E. Leonardi, M. Mellia, F. Neri, M. Ajmone Marsan, "Bounds on Average Delays and Queue Length Averages and Variances in Input Queued and Combined Input/Output Queued Cell-Based Switches", *Journal of the ACM*, vol. 50, n. 4, July 2003
- [17] X. Lin, P.K. McKinley, L.M. Ni, "The message flow model for routing in wormhole-routed networks", *IEEE Trans. on Parallel and Distributed Systems*, vol. 6, n. 7, July 1995, pp. 755-760
- [18] R.B. Magill, C.E. Rohrs, R.L. Stevenson, "Output queued switch emulation by fabrics with limited memory", *IEEE Journal on Selected Area in Communications*, vol. 21, n. 4, May 2003
- [19] S. Mascolo, D. Cavendish, M. Gerla, "ATM rate based congestion control using a Smith predictor: an EPRCA implementation", *IEEE INFOCOM '96*, San Francisco, CA, USA, Mar. 1996, pp. 569-576
- [20] N. McKeown, A. Mekkittikul, V. Anantharam, J. Walrand, "Achieving 100% throughput in an input-queued switch", *IEEE Trans. on Communications*, vol. 47, n. 8, Aug. 1999, pp. 1260-1272
- [21] S.P. Meyn, R. Tweedie, *Markov Chain and Stochastic Stability*, Springer-Verlag, 1993
- [22] L. Mhamdi, M. Hamdi, "MCBF: a high-performance scheduling algorithm for buffered crossbar switches", *IEEE Communications Letters*, vol. 7, n. 9, Sept. 2003, pp. 451-453
- [23] M. Nabeshima, "Performance evaluation of a combined input and crosspoint queued switch", *IEICE Trans. Comm.*, vol. E83-B, n. 3, Mar. 2000
- [24] M.J. Neely, E. Modiano, C.E. Rohrs, "Dynamic power allocation and routing for time varying wireless networks", *IEEE INFOCOM 2003*, vol. 1, Mar. 2003, pp. 745-755
- [25] M.L. Neely, E. Modiano, C.E. Rohrs "Power Allocation and Routing in Multibeam Satellites with Time-Varying Channels", *IEEE/ACM Trans. on Networking*, vol. 11, n. 1, Feb. 2003, pp. 138-152
- [26] L.M. Ni, P.K. McKinley, "A survey of wormhole routing techniques in direct networks" *IEEE Computer*, vol. 26, n. 2, Feb. 1993, pp. 62-76
- [27] R. Rojas Cessa, E. Oki, Z. Jing, H.J. Chao, "CIXB-1: combined input-one-cell-crosspoint buffered switch", *IEEE HPSR 2001*, Dallas, USA, pp. 324-329
- [28] R. Rojas Cessa, E. Oki, H.J. Chao, "CIXOB-k: combined input-crosspoint-output buffered packet switch", *IEEE GLOBECOM 2001*, Nov. 2001, pp. 2654-60
- [29] R. Rojas-Cessa, E. Oki, "Round robin selection with adaptable size frame in a combined input-crosspoint buffered switch", *IEEE Communications Letters*, vol. 7, n. 11, Nov. 2003
- [30] K. Ross, N. Bambos, "Local Search Scheduling Algorithms for Maximal Throughput in Packet Switches", *IEEE INFOCOM 2004*, Mar. 2004
- [31] L. Tassiulas, A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multi-hop radio networks", *IEEE Trans. on Automatic Control*, vol. 37, n. 12, Dec. 1992, pp. 1936-1948

- [32] L. Tassiulas, "Scheduling and performance limits of networks with constantly changing topology", *IEEE Transactions on Information Theory*, vol. 43, n. 3, May 1997, pp. 1067-1073
- [33] L. Tassiulas, "Linear complexity algorithms for maximum throughput in radio networks and input queued switches", *IEEE INFOCOM 1998*, San Francisco, CA, USA, Apr. 1998
- [34] R. Telikepalli, T. Drwiega, J. Yan, "Storage area network extension solutions and their performance assessment", *IEEE Communications Magazine*, vol. 42, n. 4, April 2004, pp. 56-63
- [35] K. Yoshigoe, K.J. Christensen, "An evolution to crossbar switches with virtual output queueing and buffered crosspoint", *IEEE Network*, Sept. 2003, pp. 48-56

APPENDIX I
PROOF OF THEOREM 2

Proof: Consider the following Lyapunov³ function: $\mathcal{L}(Y(n)) = Y(n)\mathbf{M}(n)Y^T(n)$. If $\Delta\mathcal{L}(n) = E[\mathcal{L}(Y(n+1)) - \mathcal{L}(Y(n))|Y(n)]$, the stability criteria of (8) becomes:

$$\lim_{\|Y(n)\| \rightarrow \infty} \frac{\Delta\mathcal{L}(n)}{\|Y(n)\|} < -\epsilon \quad (16)$$

where

$$\begin{aligned} \Delta\mathcal{L}(n) &= E[Y(n+1)\mathbf{M}(n+1)Y(n+1)^T - \\ &Y(n)\mathbf{M}(n)Y(n)^T] = E[Y(n)\mathbf{M}(n+1)Y(n)^T + \\ &2[A(n) - S(n)(\mathbf{I} - \mathbf{R})]\mathbf{M}(n+1)Y^T(n) + \\ &[A(n) - S(n)(\mathbf{I} - \mathbf{R})]\mathbf{M}(n+1)[A(n) - S(n)(\mathbf{I} - \mathbf{R})]^T - \\ &Y(n)\mathbf{M}(n)Y(n)^T] \quad (17) \end{aligned}$$

Now observe the $\mathbf{M}(n+1) = \mathbf{M}(n) + o(\|Y\|)$ when $\|Y\| \rightarrow \infty$, since $|y_{f(q)}(n+1) - y_{f(q)}(n)| \leq 1$. Hence, we can substitute $\mathbf{M}(n+1)$ with $\mathbf{M}(n)$ and obtain:

$$\begin{aligned} \Delta\mathcal{L}(n) &= \\ &E[2[A(n) - S(n)(\mathbf{I} - \mathbf{R})]\mathbf{M}(n)Y^T(n)] + \\ &E[A(n) - S(n)(\mathbf{I} - \mathbf{R})]\mathbf{M}(n) \\ &[A(n) - S(n)(\mathbf{I} - \mathbf{R})]^T] = \\ &2E[[\Lambda - S(n)(\mathbf{I} - \mathbf{R})]\mathbf{M}(n)Y^T(n)] + \\ &E[A(n) - S(n)(\mathbf{I} - \mathbf{R})]\mathbf{M}(n)[A(n) - S(n)(\mathbf{I} - \mathbf{R})]^T \quad (18) \end{aligned}$$

From now on, for the sake of readability, we will omit the variable n from our notations, when not necessary. Now consider the second term in second adder in (18):

$$\begin{aligned} &E[[A - S(\mathbf{I} - \mathbf{R})]\mathbf{M}[A - S(\mathbf{I} - \mathbf{R})]^T] = \\ &E[AM A^T - 2AM[S(\mathbf{I} - \mathbf{R})]^T + \\ &S(\mathbf{I} - \mathbf{R})\mathbf{M}[S(\mathbf{I} - \mathbf{R})]^T] \quad (19) \end{aligned}$$

Since $AM = A$, the first and second terms in (19) are negligible with respect to $\|Y\| \rightarrow \infty$. Indeed:

$$\Lambda\mathbf{M}[S(\mathbf{I} - \mathbf{R})]^T = \Lambda[S(\mathbf{I} - \mathbf{R})]^T = S(\mathbf{I} - \mathbf{R})\Lambda^T$$

and

$$[(\mathbf{I} - \mathbf{R})\Lambda^T]_q = \begin{cases} \lambda_q - \lambda_{p(q)} = \lambda_q & \text{if } q \in \Phi_I \\ 0 & \text{if } q \in \Phi_M \end{cases}$$

³Note that $\mathcal{L}(Y(n)) \geq 0$ and $\mathcal{L}(Y_0) = 0$ if Y_0 is the null vector.

$$S(\mathbf{I} - \mathbf{R})\Lambda^T = \sum_{q \in \Phi_I} d_q \lambda_q$$

which is $o(\|Y\|)$. Because of the two negligible terms, (19) becomes equal to $f_2(S)$. Now (18) becomes:

$$\Delta\mathcal{L} \approx 2\Lambda\mathbf{M}Y^T - 2f_1(S) + f_2(S) \quad (20)$$

If we now define $\Gamma = \Lambda(\mathbf{I} - \mathbf{R})^{-1}$, then $\Lambda\mathbf{M}Y^T$ can be written as $f_1(\Gamma)$. Since Λ is admissible it results: $\|\Gamma\| = \rho < 1$; we can now define $\hat{\Gamma}$ such that $\Gamma = \rho\hat{\Gamma}$: $\|\hat{\Gamma}\| = 1$ and $\Gamma \in \mathcal{D}$. Since f_1 is a linear function, then $f_1(\Gamma) = f_1(\rho\hat{\Gamma}) = \rho f_1(\hat{\Gamma})$. Now $2\rho f_2(\hat{\Gamma}) = 2\rho^{-1}f_2(\Gamma) = 2\rho^{-1}\Lambda\mathbf{M}\Lambda^T$ is $o(\|Y\|)$ and can be subtracted from (20):

$$\begin{aligned} \Delta\mathcal{L} &\approx 2f_1(\Gamma) - f(S) = 2\rho f_1(\hat{\Gamma}) - f(S) \approx \\ &2\rho f_1(\hat{\Gamma}) - 2\rho f_2(\hat{\Gamma}) - f(S) = \rho f(\hat{\Gamma}) - f(S) \end{aligned}$$

Now, considering the definition of policy \mathcal{P}_1 , $f(S) \geq f(\hat{\Gamma})$ and we can say:

$$\Delta\mathcal{L} \leq \rho f(\hat{\Gamma}) - f(\hat{\Gamma}) = -(1 - \rho)f(\hat{\Gamma}) \quad (21)$$

By neglecting terms $o(\|Y\|)$:

$$f(\hat{\Gamma}) = 2f_1(\hat{\Gamma}) = \frac{2}{\rho}f_1(\Gamma) = \frac{2}{\rho} \sum_{q \in \Phi_I} \lambda_q y_q \geq \frac{2}{\rho} \lambda_{\min} \sum_{j=1}^J y_{j,1}$$

where $\lambda_{\min} = \min_{q \in \Phi_I} \{\lambda_q : \lambda_q > 0\}$. Observe now that $\|Y\| \leq \sum_{j=1}^J y_{j,1} + \sum_{j=1}^J h_j l_j$, which can be restated as: $\|Y\| \leq \sum_{j=1}^J y_{j,1} + o(\|Y\|)$. Hence, $f(\hat{\Gamma}) \geq 2\lambda_{\min}\|Y\|/\rho$ for $\|Y\| \rightarrow \infty$. (21) becomes:

$$\Delta\mathcal{L} \leq -2\frac{1-\rho}{\rho}\lambda_{\min}\|Y\|$$

and this implies that, for any $l_j \geq 1$,

$$\lim_{\|Y\| \rightarrow \infty} \frac{\Delta\mathcal{L}}{\|Y\|} < -2\frac{1-\rho}{\rho}\lambda_{\min}$$

■

APPENDIX II
PROOF OF THEOREM 3

Proof: Consider again the Lyapunov function: $\mathcal{L}(X) = X\mathbf{M}X^T$. Eq. (18) still holds:

$$\begin{aligned} \Delta\mathcal{L} &= 2[\Lambda - D(\mathbf{I} - \mathbf{R})]\mathbf{M}X^T + \\ &E[A - D(\mathbf{I} - \mathbf{R})]\mathbf{M}[A - D(\mathbf{I} - \mathbf{R})]^T \quad (22) \end{aligned}$$

Now consider the specific policy \mathcal{P}_2 running. $D(n)$ is selected, according to (14), on the set of all possible service vectors Z such that $\|Z\| \leq 1$. If we choose Z as: $Z = \Lambda(\mathbf{I} - \mathbf{R})^{-1} + (1 - \rho)U$ with any U such that $\|U\| = 1$, then $\|Z\| \leq 1$, since: $\|Z\| = \|\Lambda(\mathbf{I} - \mathbf{R})^{-1} + (1 - \rho)U\| \leq \|\Lambda(\mathbf{I} - \mathbf{R})^{-1}\| + \|(1 - \rho)U\| = \rho + (1 - \rho) = 1$. Now:

$$\begin{aligned} D(\mathbf{I} - \mathbf{R})\mathbf{M}X^T &\geq \\ &[\Lambda(\mathbf{I} - \mathbf{R})^{-1} + (1 - \rho)U](\mathbf{I} - \mathbf{R})\mathbf{M}X^T = \\ &\Lambda\mathbf{M}X^T + (1 - \rho)U(\mathbf{I} - \mathbf{R})\mathbf{M}X^T \quad (23) \end{aligned}$$

Thanks to (23), we can bound the first adder in (22):

$$\begin{aligned} [\Lambda - D(\mathbf{I} - \mathbf{R})]\mathbf{M}\mathbf{X}^T &\leq \Lambda\mathbf{M}\mathbf{X}^T - \\ \Lambda\mathbf{M}\mathbf{X}^T - (1 - \rho)U(\mathbf{I} - \mathbf{R})\mathbf{M}\mathbf{X}^T &= \\ - (1 - \rho)UW^T \end{aligned} \quad (24)$$

The second term in (22) can be treated as the second term in (18). If we now evaluate (22), by combining (24) and (11), we can write:

$$\begin{aligned} \Delta\mathcal{L} &\leq -2(1 - \rho) \sum_{q=1}^Q u_q w_q + \\ &\sum_{q \in \Phi_M} (d_q - d_{u(q)})^2 \frac{x_{f(q)}}{l_{j(q)}} \end{aligned} \quad (25)$$

Now let $V = \mathbb{I}/\|\mathbb{I}\| \in \mathcal{D}$; it results:

$$\begin{aligned} \Delta\mathcal{L} &\leq -2(1 - \rho) \frac{1}{\|\mathbb{I}\|} \sum_{q=1}^Q w_q + \sum_{q \in \Phi_M} (d_q - d_{u(q)})^2 \frac{x_{f(q)}}{l_{j(q)}} = \\ &= -\frac{2(1 - \rho)}{\|\mathbb{I}\|} \sum_{j=1}^J x_{j,1} + \sum_{j=1}^J \frac{x_{j,1}}{l_j} \sum_{h=2}^{h_j} (d_{j,h} - d_{j,h-1})^2 \leq \\ &\quad - \sum_{j=1}^J x_{j,1} \left[\frac{2(1 - \rho)}{\|\mathbb{I}\|} - \frac{h_j - 1}{l_j} \right] \end{aligned} \quad (26)$$

where we exploited the fact that, for any j , thanks to the telescopic sum:

$$\begin{aligned} \sum_{h=1}^{h_j} w_{j,h} &= \frac{x_{j,1}}{l_j} [(l_j - x_{j,2}) + \\ &\sum_{h=2}^{h_j-1} (x_h - x_{h+1}) + x_{h_j}] = x_{j,1} \end{aligned}$$

Furthermore, $\sum_{h=2}^{h_j} (d_{j,h} - d_{j,h-1})^2 \leq h_j - 1$.

As a consequence, a sufficient condition to make the Lyapunov function drift negative is:

$$\frac{2(1 - \rho)}{\|\mathbb{I}\|} - \frac{h_j - 1}{l_j} > 0$$

which implies:

$$l_j > \frac{(h_j - 1)\|\mathbb{I}\|}{2(1 - \rho)} \quad \forall j$$

■

APPENDIX III PROOF OF COROLLARY 1

Proof: In this case from:

$$\begin{aligned} \Delta\mathcal{L} &\leq -2(1 - \rho) \left[\sum_{q \in \Phi_I} d_q x_q \left(1 - \frac{x_{p(q)}}{l_{j(q)}} \right) + \right. \\ &\sum_{q \in \Phi_M} d_q x_{f(q)} \left(\frac{x_q - x_{p(q)}}{l_{j(q)}} \right) \left. \right] + \\ &\sum_{q \in \Phi_M} (d_q - d_{u(q)})^2 \frac{x_{f(q)}}{l_{j(q)}} \end{aligned} \quad (27)$$

which, substituting D to U , becomes:

$$\begin{aligned} \Delta\mathcal{L} &\leq -2(1 - \rho) \sum_{j=1}^J \frac{x_{j,1}}{l_j} [d_{j,1}(l_j - x_{j,2}) + \\ &d_{j,2}x_{j,2}] + \sum_{j=1}^J \frac{x_{j,1}}{l_j} (d_{j,1} + d_{j,2} - 2d_{j,1}d_{j,2}) \\ &\leq -2(1 - \rho) \sum_{j=1}^J \frac{x_{j,1}}{l_j} \left[d_{j,1} \left(l_j - x_{j,2} - \frac{1}{2(1 - \rho)} \right) + \right. \\ &\quad \left. d_{j,2} \left(x_{j,2} - \frac{1}{2(1 - \rho)} \right) \right] \end{aligned} \quad (28)$$

in which from (14):

$$D = \arg \max_{D \in \mathcal{D}} \sum_{j=1}^J \frac{x_{j,1}}{l_j} [d_{j,1}(l_j - x_{j,2}) + d_{j,2}x_{j,2}]$$

A sufficient condition which ensures for any the Lyapunov function drift to be negative, is:

$$\frac{1}{2(1 - \rho)} < 1$$

Indeed, $d_{j,1} = 1$ only when $l_j - x_{j,2} \geq 1$ and $d_{j,2} = 1$ only when $x_{j,2} \geq 1$. ■