Joint Optimal Scheduling and Routing for Maximum Network Throughput

Emilio Leonardi, Marco Mellia, Marco Ajmone Marsan, Fabio Neri
Dipartimento di Elettronica,
Politecnico di Torino, Italy
E-mail: {leonardi, mellia, ajmone, neri} @mail.tlc.polito.it.

Abstract— **In this paper we consider packet networks loaded by admissible traffic patterns, i.e. by traffic patterns that, if optimally routed, do not overload network resources. In these conditions, we study the combined behavior of distributed dynamic routing and scheduling algorithms based upon link state information, with no knowledge of the average traffic pattern, and we prove that simple schemes can achieve the same network throughput as optimal centralized routing and scheduling algorithms with complete information on the traffic pattern.**

Our study is based on a *flow-level* abstract model of the network, and considers *elastic* traffic, i.e., we assume that flows can adapt their transmission rates to network conditions. As a result, our model captures some of the main features of Internet traffic and of quality-of-service routing approaches being currently proposed for IP networks.

We show that efficient dynamic routing and scheduling algorithms can be implemented in a distributed way, and we prove that maximum throughput is achieved also in case of temporary mismatches between the actual link metrics and those used by the routing algorithm. This is a particularly relevant aspect, since any distributed implementation of a routing algorithm requires a periodic exchange of link state information among nodes, and this implies delays, and thus time periods in which the actual link state is not known.

I. Motivation and Previous Work

In the early years of packet networks, a hot research topic was the definition of efficient *dynamic* routing algorithms, aiming at an optimal exploitation of network resources by adapting packet routes to the instantaneous traffic conditions. Dynamic routing algorithms can in principle offer significant advantages with respect to static routing, since they automatically react to congestion situations, therefore offering better performance and quality of service (QoS). However, the finding that dynamic routing algorithms may lead to route flapping and to the consequent instabilities and performance degradations, has limited their diffusion. As a consequence, the dynamic features implemented today in routing protocols are mostly limited to automatic reactions to topology changes due to link failures or infrastructure updates.

In recent years, dynamic routing algorithms have again attracted the attention of the networking community. Several aspects have been investigated, such as: i) protocol convergence [1], [2], ii) overhead impact [3], [4], iii) implementation issues [5], iv) impact of update policies [3], and v) performance issues.

The last topic is of particular interest, and has often been associated with QoS routing. Indeed, the core of any routing algorithm is a state-dependent cost function that is used to find the optimal (or at least a good) route across the network by solving an optimization problem. In particular, given the best-effort nature of the current Internet, where the majority of data flows are elastic (i.e., they can adapt to network conditions), the most commonly used optimization metric aims either at the maximization of user throughput, or at the minimization of delays. Several recent QoS routing proposals introduced dynamic algorithms and protocols that provide advantages over traditional, topology-based algorithms, such as the shortest path routing presently used in TCP/IP networks [6], [7], [8], [9], [10], [11].

Within QoS routing studies, two main research areas can be identified. The first area, sometimes also called "traffic engineering", considers as input to the design of routing algorithms the information about the traffic pattern that users offer to the network. We call this class of algorithms "traffic-aware" QoS routing algorithms. Given the knowledge of the network topology, and a utility function, the goal of a traffic-aware routing algorithm is to find a set of routes that maximizes this utility function, under technological or performance constraints. Traffic-aware QoS routing algorithms can be formalized as Linear Programming (LP) problems, and efficient solutions are available since the early seventies [12], [13]. The result of the optimization is, for each traffic relation, a set of paths and a corresponding set of probabilities that specifies the traffic splitting over each path. The implementation of such a routing algorithm in the Internet can be obtained either by using explicit path mechanisms, such as those offered by MPLS [14], or by exploiting the recent results in [6], [7], which show that with the current setup of IP networks it is possible to find: i) an equivalent set of link costs which implements the same routes as those of an explicit path mechanism, and ii) a set of rules to implement the same traffic splitting. The difficulty in estimating the traffic pattern and in solving the LP problem makes these algorithms suitable for a static and centralized implementation. For this reason, in the following we refer to them as *static routing* algorithms.

The second area within QoS routing research assumes that the traffic pattern is not known. We call the corresponding class of algorithms "traffic-unaware" QoS routing. In this case, the optimization of routing is based on direct measurements of the

0-7803-8968-9/05/$20.00 (c)2005 IEEE

performance of network elements, like link utilizations, queue lengths, etc. Given the current state of the network, the goal of a traffic-unaware QoS routing algorithm is to find a set of paths that can be used to accommodate traffic requests. Among the proposed heuristics, the "Widest-Shortest" [8] and the "Minimum-Distance" [9] approaches are generally considered to be good routing algorithms. These findings have been generalized by recent results that consider resilience to traffic load variations [10], and more general cost functions [11]. The characteristics of traffic-unaware routing algorithms favor a distributed implementation in which link costs are dynamically advertised, and routes updated to follow the instantaneous traffic variations. For this reason, in the following we refer to them as *dynamic routing* algorithms.

The performance evaluation of dynamic (traffic-unaware) QoS routing algorithms was traditionally carried out by simulation only. Thus, to the best of our knowledge, no analytical results were originally derived to characterize their properties, and in particular to define the network stability region when these algorithms are used.

The recent introduction of powerful theoretical approaches in network studies, such as the stochastic Lyapunov function methodology [15], [16], opened new perspectives to theoretical studies on the impact of routing algorithms. Using this approach, for example, the authors of [17] considered adhoc wireless networks, and defined a dynamic scheduling policy that maximizes the stability region for a given static routing algorithm. Recently, the authors of [18] extended the results of [17] defining joint packet-by-packet dynamic routing algorithm and power allocation schemes, which maximize the network stability region. Finally, the authors of [19] consider a generic queueing system and derive stability conditions under more general traffic patterns than the one originally considered in [17].

In this paper, we consider a more general and abstract flow-level network model, which allows us on one side to consider the intrinsic elasticity of the traffic, and on the other side to almost completely neglect the packet-level dynamics, thus greatly reducing the complexity of the system under study. Using this model, we analyze the stability properties of different classes of dynamic (traffic-unaware) routing algorithms and bandwidth allocation policies.

In particular, we consider packet networks loaded by admissible traffic patterns, i.e. traffic patterns that, if optimally routed, do not overload network resources. Under these conditions, we prove that simple distributed dynamic routing and scheduling algorithms only based upon link state information can achieve the same network throughput as optimal centralized routing and scheduling algorithms with complete information on the traffic pattern.

We also show that such dynamic routing and scheduling algorithms can be implemented in a distributed way, and we prove that maximum throughput is achieved also in case of temporary mismatches between the ideal link costs and those used by the routing algorithm. This is a particularly relevant aspect, since any distributed implementation of a routing algorithm requires a periodic exchange of link state information among nodes, and this implies delays, and thus time periods in which the updated state of links is not known.

Our proof of the equivalence of the maximum throughput achievable by static and dynamic routing algorithms was inspired by the well-known result about the equivalence in the maximum throughput achievable with input queued and output queued switches [20], and it may open interesting new possibilities for the adoption of dynamic routing algorithms.

Even if our contribution is mostly theoretical, some of the proposed dynamic routing and scheduling algorithms are compatible with a practical implementation in a real network scenario.

The rest of this paper is organized as follows. Section II presents our modeling assumptions. Section III introduces the traffic admissibility and sustainability definitions, and recalls the stability criterion based on the stochastic Lyapunov function that is used in later sections. Section IV presents preliminary results, relating to an idealized scenario. The proof is also extended to the case of a finite error in the link cost estimation. Section V presents our main results, proving that, under any admissible traffic pattern, a network adopting simple scheduling and routing algorithms can achieve the same network throughput of an optimal static routing algorithm. Section VI particularizes the result to the case of exponentially distributed flow sizes, while Section VII discusses further generalizations. Finally, Section VIII discusses implementation issues, and Section IX concludes the paper.

II. Modeling Assumptions and Notation

A network is represented by a directed graph $G = (V, E)$ where V is the set of vertexes (which represent the network switching nodes), and E is the set of edges (which represent the network transmission links). $C^l = C^{l(i \to j)},\ l \in E,\ i, j \in V$ denotes the capacity of link l, which connects node i to node j. To simplify notation, we assume all links to be of the same capacity: $C^l = C = 1,\ \forall l \in E$. However, this assumption can be easily relaxed, as we will show at the end of the paper in a special case.

The traffic in the network is modeled at the level of *flows*, neglecting individual packets. Elastic flows arrive at the network according to a Poisson process; each flow is associated with a source $s \in V$ and a destination $d \in V$[1], as well as a size, which represents the amount of information (in bits, bytes, or any other unit) that has to be transferred. Flow sizes are assumed to be independent random variables, distributed according to a general distribution with mean $1/\mu$, and finite polynomial moments (indeed, only the finiteness of the first two moments suffices when the quadratic Lyapunov function is adopted – see below).

Each flow is routed through the network according to a given algorithm, and it remains active until its last bit is successfully delivered to the destination. During its active

[1] Flows are considered to be unidirectional. Bidirectional flows translate into two unidirectional flows. Correlations between flow arrivals do not impair our modeling assumptions.

period, each flow receives a portion of the network capacity, which varies with the network congestion (for example, with the number of concurrent flows), i.e., flow sources can adapt their sending rate to network congestion; the link capacity is allocated to flows according to a scheduling policy, which may derive either from the end-to-end congestion control mechanism implemented at source nodes, like with the TCP congestion control, or from the network nodes, like with GPS scheduling policies [21].

Each link l is modeled as a flow-level queue. The queue $q^l = q^{l(i \to j)},\ l \in E,\ i, j \in V$ corresponds to link l, which connects node i to node j.

Let λ_{sd} be the average flow arrival rate for each source-destination pair; $\rho_{sd} = \lambda_{sd}/\mu$ is the average workload associated with flows from s to d. The traffic pattern in the network is described by the matrix $\boldsymbol{\rho} \ = [\rho_{sd}]$, which is called the traffic matrix.

As soon as a given flow from s to d arrives at the network, it is routed over a path, call it π, which comprises several links. All packets of that flow are transferred over path π.

Let $x^l(n)$ represent the virtual backlog at link l; i.e., $x^l(n)$ represents the total amount of information in bits associated with active flows already routed through link l, that at time n must still traverse link l. Let[2] $\mathbf{X}(n) \in \mathbb{N}^{|E|}$ be an $|E|$-dimensional vector[3], whose components are the $x^l(n)$. Note that the virtual backlog at time n does not necessarily equal the physical backlog, i.e., the amount of information physically enqueued at queue l at time n, since some information may be physically queued at previous links along the path.

In order to simplify the notation, we consider discrete-time systems, even if all results can be easily generalized to continuous-time systems. The discrete-time steps can be seen as corresponding to equally spaced snapshots of the continuous system state; the scheduling and routing decisions are updated in correspondence with these snapshots.

The routing algorithm is used to associate a path with each new flow that arrives to the network. A **static** routing algorithm is defined by matrix $\hat{\mathbf{R}} \in \mathbb{R}_+^{|V|^2 \times |E|}$, whose element $\hat{r}^l_{sd}$ is the percentage of flows from source s to destination d that, according to routing algorithm $\hat{\mathbf{R}}$, is routed through link l. Since a static routing algorithm is traffic-aware, $\hat{\mathbf{R}}$ is a function of traffic matrix $\boldsymbol{\rho}$, $\hat{\mathbf{R}} = \hat{\mathbf{R}}(\boldsymbol{\rho})$

Given any source-destination pair, let $\Pi(sd)$ be the set of possible paths from source s to destination d, and $\Pi = \bigcup_{s,d} \Pi(sd)$ be the set of all possible paths. Let $L(\pi)$ be the set of links (i.e., queues) traversed by path $\pi \in \Pi(sd)$. Let $\Pi(l)$ be the set of paths (from any source to any destination) traversing link l. An equivalent representation of the routing algorithm $\hat{\mathbf{R}}$ can be given by assigning the set of values $\hat{r}^\pi_{sd}$ which represent the percentage of $s \to d$ flows that, according to routing algorithm $\hat{\mathbf{R}}$, is routed through path π. We define accordingly the path routing matrix $\hat{\mathbf{R}}^\pi \in \mathbb{R}_+^{|V|^2 \times |\Pi|}$.

Notice that it is always possible to decompose the traffic routed over link l as the sum of the traffic routed over paths π that use link l; indeed:

$$\hat{r}^l_{sd} = \sum_{\pi \in \Pi(l) \cap \Pi(sd)} \hat{r}^\pi_{sd} \qquad (1)$$

A **dynamic** routing algorithm is defined by a sequence of matrices $\mathbf{R}(n)$. The element $r^l_{sd}(n)$ is the percentage of flows from s to d that, according to the dynamic routing algorithm, is routed through link l at time n. We consider dynamic routing algorithms in which $\mathbf{R}(n)$ is a function of the network state $\mathbf{X}(n)$, $\mathbf{R}(n) = \mathbf{R}(\mathbf{X}(n))$. An equivalent representation of the routing algorithm $\mathbf{R}$ can be given with the path routing matrix $\mathbf{R}^\pi(n)$, whose elements $r^\pi_{sd}(n)$ represent the percentage of flows from s to d routed over path π at time n.

III. Traffic Admissibility and Sustainability

We say that a traffic matrix $\boldsymbol{\rho}$ is admissible (and the corresponding traffic pattern is admissible) if there exists a static routing algorithm $\hat{\mathbf{R}}$ under which all the queues (links) in the network are not overloaded. More formally:

Definition 1: A traffic pattern is *admissible* if there exists a *static* routing algorithm $\hat{\mathbf{R}}$, such that

$$\sum_{sd} \hat{r}^l_{sd} \rho_{sd} < 1 \qquad \forall l \in E \qquad (2)$$

The problem of finding the static routing algorithm $\hat{\mathbf{R}}$ can be formalized as a Linear Programming problem. Let $f^{l(i \to j)}_{sd} = \hat{r}^l_{sd} \rho_{sd}$ be variables which represent the average normalized amount of information from s to d that is routed through link l from i to j. Variables $f^{l(i \to j)}_{sd}$ must satisfy

$$\sum_j f^{l(s \to j)}_{sd} = \rho_{sd} \qquad \forall s, d \qquad (3)$$

$$\sum_j f^{l(i \to j)}_{sd} - \sum_j f^{l(j \to i)}_{sd} = 0 \qquad \forall i,\ s \neq i, d \neq i \quad (4)$$

$$\sum_i f^{l(i \to d)}_{sd} = \rho_{sd} \qquad \forall s, d \qquad (5)$$

$$\sum_{sd} f^{l(i \to j)}_{sd} < 1 \qquad \forall l \qquad (6)$$

$$f^{l(i \to j)}_{sd} \geq 0 \qquad \forall l, s, d \qquad (7)$$

Equations (3), (4), (5) represent the classic flow conservation equations; equation (6) forces all the flows routed over a link to be admissible, and finally equation (7) forces all the flows to be non-negative. By selecting an appropriate utility function, the above expressions can be mapped into an optimization problem which defines the *optimal* routing algorithm under the chosen utility function. This class of routing algorithms belongs to the static class [13], [6], [7], and it assumes the knowledge of the traffic matrix $\boldsymbol{\rho}$.

Now we introduce the important definition of traffic pattern sustainability: a traffic pattern is sustainable if the network of queues can be kept stable. Sustainability depends on the scheduling policy, i.e., on the way in which the different flows queued at the same queue are served. More formally:

[2] We denote with $\mathbb{N}$ ($\mathbb{N}_+$) the set of (non-negative) integers, and with $\mathbb{R}$ ($\mathbb{R}_+$) the set of (non-negative) real numbers.

[3] We use column vectors throughout the paper.

Definition 2: A traffic pattern $\boldsymbol{\rho}$ is *sustainable* if there exist a static routing algorithm $\hat{\mathbf{R}}$ and a scheduling policy such that:

$$\limsup_{n\to\infty} E\left[||\mathbf{X}(n)||\right] < \infty$$

where $||\cdot||$ is any norm function.

The main difference between admissibility and sustainability is the fact the former is defined on the average load, and requires that no network element is overloaded, while the latter requires that network elements are capable of providing service in order to avoid congestion build up. It is immediate to verify that admissibility is a necessary condition for a traffic pattern to be sustainable; the reverse statement is, instead, less evident. It was indeed proven that aggressive scheduling policies may lead to instabilities even in presence of admissible traffic. See for example [22].

Since sustainability is related to scheduling policies, which operate on the physical backlog at network queues, we can observe that, when the physical and the virtual backlogs are the same, admissibility implies sustainability. We will show in the next sections that, under our assumptions, and with scheduling policies related to link loads, the two definitions of admissibility and sustainability become equivalent.

A. Lyapunov Stability Criterion

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 [15] and [16] for more details.

Theorem 1: Let $\mathbf{X}(n)$ be a $|E|$-dimensional Markov chain, whose elements $x^l(n)$ are non-negative integers, i.e., $\mathbf{X}(n) \in \mathbb{N}_+^{|E|}$. If there exists a non-negative valued function $\{\mathcal{L} : \mathbb{N}_+^{|E|} \to \mathbb{R}_+\}$ such that:

$$E[\mathcal{L}(\mathbf{X}(n+1)) - \mathcal{L}(\mathbf{X}(n)) \mid \mathbf{X}(n)] < \infty \qquad (8)$$

$$\limsup_{||\mathbf{X}(n)||\to\infty} \frac{E[\mathcal{L}(\mathbf{X}(n+1)) - \mathcal{L}(\mathbf{X}(n)) \mid \mathbf{X}(n)]}{||\mathbf{X}(n)||} < -\epsilon \qquad (9)$$

for some $\epsilon > 0$, then $\mathbf{X}(n)$ is positive recurrent, and

$$\limsup_{n\to\infty} E[||\mathbf{X}(n)||] < \infty$$

Inequality (8) imposes that the increments of the Lyapunov function $\mathcal{L}(\mathbf{X})$ be finite. It is immediate to verify that this constraint can be met in general if all the moments of $\mathbf{X}$ are finite; in particular, for quadratic Lyapunov functions, it is sufficient that the second moment of $\mathbf{X}$ is finite. The second inequality, (9), often termed as the Lyapunov function drift, requires that, for large values of $||\mathbf{X}||$, the average increment in the Lyapunov function from time n to time $n{+}1$ be 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.

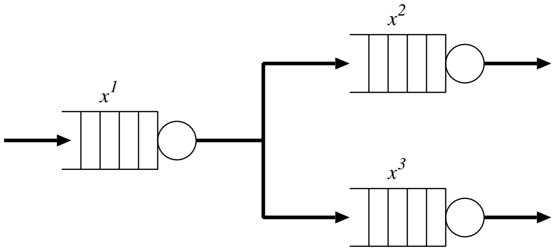

Fig. 1. A simple network topology in which some links cannot be fully loaded.

IV. Results for an Optimistic Scenario

The results presented in this section are preliminary, since they refer to an ideal, optimistic scenario, in which we assume that the capacity of every link $l \in E$ is fully utilized, as long as the link virtual backlog $x^l(n)$ is non null.

This is not true in general, as it can be understood by considering the simple example shown in Figure 1. Remember that we are considering a flow-level model, and that the link virtual backlog $x^l(n)$ corresponds to the total amount of information associated with active flows already routed through link l, that at time n must still traverse link l. Assume that for the three links in the figure the virtual backlogs are non null. From the figure we immediately understand that link 1 is the system bottleneck: if link 1 works at its capacity, it cannot feed enough data into links 2 and 3 to keep them fully utilized.

Nevertheless, with our optimistic assumption (which will be corrected in the next section) the dynamics of the virtual backlog on link l is given by the following equation:

$$x^l(n+1) = \max(x^l(n) - 1, 0) + \sum_{sd} r^l_{sd}(n) a_{sd}(n)$$

where we assume a dynamic routing algorithm, and $a_{sd}(n)$ represents the workload provided by flows with origin s and destination d arrived during time slot n; we notice that $\rho_{sd} = E[a_{sd}]$.

We use a vectorial notation, in which scalar operators are extended to vectors by applying the operator on all components. For example, given two vectors $\mathbf{A}$ e $\mathbf{B}$, the vector $\max(\mathbf{A}, \mathbf{B})$ contains components $\max(a^l, b^l)$. We can write:

$$\mathbf{X}(n+1) = \max(\mathbf{X}(n) - \mathbb{1}, \mathbf{0}) + \mathbf{R}^T(n)\mathbf{A}(n) \qquad (10)$$

where $\mathbf{X}(n)$ is the vector of virtual backlogs; $\mathbb{1}$ is an $|E|$-dimensional vector with all entries equal to 1; $\mathbf{A}(n) \in \mathbb{R}^{|V|^2}$ is the vector whose components represent the workloads provided by flows arrived during time slot n; i.e., $\mathbf{A}(n) = [a_{sd}(n)]$.

Equation (10) implies that the system dynamics are described by an irreducible Markov chain whose state descriptor is $\mathbf{X}$.

Under these idealized conditions, it is immediate to realize that the sustainability and admissibility conditions are equivalent. Indeed, any scheduling policy capable of guaranteeing to flows service rates equal to the average virtual backlog, i.e., to the average amount of traffic routed over the different links, ensures queue boundedness to any admissible traffic.

The question we raise now is the following: "Is it possible to devise a dynamic routing algorithm that can guarantee stability for any admissible traffic pattern, e.g., without requiring the knowledge of the traffic matrix $\boldsymbol{\rho}$?"

Our answer is positive; the following theorem defines a simple dynamic routing algorithm that guarantees stability under any admissible traffic pattern. We call this algorithm *minimum-backlogged-path routing* since newly arrived flows $s \rightarrow d$ that arrive at time slot n are routed according to a minimum cost path strategy in which link costs equal the link virtual backlogs. That is, the algorithm selects the path $\pi \in \Pi(sd)$ that minimizes the sum of virtual backlogs, i.e.,

$$\pi = \arg \min_{\pi \in \Pi(sd)} \sum_{l \in L(\pi)} x^l(n)$$

Theorem 2: A dynamic routing algorithm $\mathbf{R}^*(n)$ defined as

$$\mathbf{R}^*(n) = \arg\min_{\mathbf{R}} \mathbf{A}^T(n)\mathbf{R}\mathbf{X}(n) \tag{11}$$

stabilizes the network under any admissible traffic pattern.

Proof: We will prove this result by applying the Lyapunov function methodology. Let us consider the quadratic Lyapunov function

$$\mathcal{L}(\mathbf{X}) = \mathbf{X}^T\mathbf{X} \tag{12}$$

Recalling that we consider Poisson arrivals of flows, and finite variance flow sizes, Inequality (8) holds. Then, consider the Lyapunov function drift (9):

$$\begin{aligned}
&\frac{E[\mathcal{L}(\mathbf{X}(n+1)) - \mathcal{L}(\mathbf{X}(n)) \mid \mathbf{X}(n)]}{||\mathbf{X}(n)||} = \\
&= \frac{E[\mathbf{X}^T(n+1)\mathbf{X}(n+1) - \mathbf{X}^T(n)\mathbf{X}(n) \mid \mathbf{X}(n)]}{||\mathbf{X}(n)||} \\
&= \frac{E\left[\max(\mathbf{X}(n) - \mathbb{1}, \mathbf{0})^T \max(\mathbf{X}(n) - \mathbb{1}, \mathbf{0})\right]}{||\mathbf{X}(n)||} \\
&\quad + \frac{2E[\mathbf{A}^T(n)\mathbf{R}^*(n)\mathbf{X}(n)] + o(||\mathbf{X}(n)||)}{||\mathbf{X}(n)||} \\
&\quad - \frac{E[\mathbf{X}^T(n)\mathbf{X}(n) \mid \mathbf{X}(n)]}{||\mathbf{X}(n)||}
\end{aligned}$$

where $\max(\mathbf{X}(n) - \mathbb{1}, \mathbf{0})$ was approximated with $\mathbf{X}(n) + o(||\mathbf{X}(n)||)$ in the second term, and terms not comprising $\mathbf{X}(n)$ were neglected at the numerator [since we consider large values of $||X(n)||$ in (9)].

Observing that $\max(\mathbf{X} - \mathbb{1}, \mathbf{0}) = \mathbf{X} - \min(\mathbf{X}, \mathbb{1})$,

$$\begin{aligned}
&E\left[\max(\mathbf{X}(n) - \mathbb{1}, \mathbf{0})^T \max(\mathbf{X}(n) - \mathbb{1}, \mathbf{0})\right] = \\
&= \mathbf{X}^T(n)\mathbf{X}(n) - 2\min(\mathbf{X}^T(n), \mathbb{1}^T)\mathbf{X}(n) \\
&\quad + \min(\mathbf{X}(n)^T, \mathbb{1}^T)\min(\mathbf{X}(n), \mathbb{1}) \\
&= \mathbf{X}^T(n)\mathbf{X}(n) - 2\mathbb{1}^T\mathbf{X}(n) + o(||\mathbf{X}||)
\end{aligned}$$

since $\min(\mathbf{X}^T, \mathbb{1}^T)\mathbf{X} = \mathbb{1}^T\mathbf{X} - o(||\mathbf{X}||)$. Finally

$$\begin{aligned}
&\frac{E[\mathcal{L}(\mathbf{X}(n+1)) - \mathcal{L}(\mathbf{X}(n)) \mid \mathbf{X}(n)]}{||\mathbf{X}(n)||} = \\
&= \frac{2E[\mathbf{A}^T(n)\mathbf{R}^*(n)]\mathbf{X}(n) - 2\mathbb{1}^T\mathbf{X}(n) + o(||\mathbf{X}(n)||)}{||\mathbf{X}(n)||}
\end{aligned} \tag{13}$$

Stability is guaranteed if there exist $\epsilon > 0$ such that the Lyapunov function drift is strictly negative for large $||\mathbf{X}(n)||$:

$$\frac{\left(E[\mathbf{A}^T(n)\mathbf{R}^*(n)] - \mathbb{1}^T\right)\mathbf{X}(n)}{||\mathbf{X}(n)||} < -\epsilon \tag{14}$$

To prove that the drift is negative, we first define the policy $\mathbf{R}'(\mathbf{X}(n)) = \arg\min_{\mathbf{R}} E[\mathbf{A}^T(n)]\mathbf{R}\mathbf{X}(n)$; note that

$$\mathbf{A}^T(n)\mathbf{R}^*(n)\mathbf{X}(n) = \min_{\mathbf{R}} \mathbf{A}^T(n)\mathbf{R}\mathbf{X}(n) \leq \mathbf{A}^T(n)\mathbf{R}'(n)\mathbf{X}(n)$$

thus

$$E[\mathbf{A}^T(n)\mathbf{R}^*(n)]\mathbf{X}(n) \leq E[\mathbf{A}^T(n)]\mathbf{R}'(n)\mathbf{X}(n)$$

Finally, since $E[\mathbf{A}^T(n)]\mathbf{R}'(n)\mathbf{X}(n) \leq E[\mathbf{A}^T(n)]\hat{\mathbf{R}}\mathbf{X}(n) < \mathbb{1}^T\mathbf{X}(n)$, where $\hat{\mathbf{R}}$ is a routing that makes $\mathbf{A}(n)$ admissible according to (2),

$$\begin{aligned}
&E[\mathbf{A}^T(n)\mathbf{R}^*(n)]\mathbf{X}(n) - \mathbb{1}^T\mathbf{X}(n) \leq \\
&\quad \leq E[\mathbf{A}^T(n)]\mathbf{R}'(n)\mathbf{X}(n) - \mathbb{1}^T\mathbf{X}(n) \\
&\quad = \min_{\mathbf{R}} E[\mathbf{A}^T(n)]\mathbf{R}\mathbf{X}(n) - \mathbb{1}^T\mathbf{X}(n) \\
&\quad \leq E[\mathbf{A}^T(n)]\hat{\mathbf{R}}\mathbf{X}(n) - \mathbb{1}^T\mathbf{X}(n) < -\epsilon
\end{aligned}$$

hence the nominator in (14) is negative. ■

The proof can be extended to the case in which the routing algorithm operates on a wrong estimate of $\mathbf{X}(n)$, as far as the mismatch between $\mathbf{X}(n)$ and its estimate $\hat{\mathbf{X}}(n)$ is on average bounded:

Theorem 3: If there exists a finite k such that $E[||\mathbf{X}(n) - \hat{\mathbf{X}}(n)|| \mid \mathbf{X}(n)] < k, \forall \mathbf{X}(n)$, then a dynamic routing algorithm $\mathbf{R}^+(n)$ defined as

$$\mathbf{R}^+(n) = \arg\min_{\mathbf{R}} \mathbf{A}^T(n)\mathbf{R}\hat{\mathbf{X}}(n) \tag{15}$$

stabilizes the network under any admissible traffic pattern.

We call this algorithm *approximate minimum-backlogged-path routing*.

Proof: We build on the same calculations and on the same Lyapunov function as in the previous proof. Stability is guaranteed if there exist $\epsilon > 0$ such that the Lyapunov function drift is strictly negative for large $||\mathbf{X}(n)||$; from (14):

$$\frac{\left(E[\mathbf{A}^T(n)\mathbf{R}^+(n)] - \mathbb{1}^T\right)\mathbf{X}(n)}{||\mathbf{X}(n)||} < -\epsilon$$

Define $\mathbf{E}(n) = \hat{\mathbf{X}}(n) - \mathbf{X}(n)$. We note that:

$$\begin{aligned}
\mathbf{A}^T(n)\mathbf{R}^+(n)\hat{\mathbf{X}}(n) &= \min_{\mathbf{R}} \mathbf{A}^T(n)\mathbf{R}(n)\hat{\mathbf{X}}(n) \\
&\leq \mathbf{A}^T(n)\mathbf{R}^*(n)[\mathbf{X}(n) + \mathbf{E}(n)]
\end{aligned}$$

Thus:

$$\begin{aligned}
&E[\mathbf{A}^T(n)\mathbf{R}^+(n)\mathbf{X}(n) \mid \mathbf{X}(n)] \\
&\quad \leq E[\mathbf{A}^T(n)\mathbf{R}^*(n)]\mathbf{X}(n) + E[\mathbf{A}^T(n)\mathbf{R}^*(n)\mathbf{E}(n) \mid \mathbf{X}(n)]
\end{aligned}$$

being the last term bounded in norm, due to the fact that $\mathbf{E}(n)$ is bounded in norm by k. As a consequence:

$$\frac{\left(E[\mathbf{A}^T(n)\mathbf{R}^{+}(n)] - \mathbb{1}^T\right)\mathbf{X}(n)}{||\mathbf{X}(n)||} \leq$$
$$\leq \frac{\left(E[\mathbf{A}^T(n)\mathbf{R}^{*}(n)] - \mathbb{1}^T\right)\mathbf{X}(n)}{||\mathbf{X}(n)||} + o(1)$$

which concludes the proof, since we have already proved that

$$\limsup_{||X(n)||\to\infty} \frac{\left(E[\mathbf{A}^T(n)\mathbf{R}^{*}(n)] - \mathbb{1}^T\right)\mathbf{X}(n)}{||\mathbf{X}(n)||} < -\epsilon$$

■

This last result is of particular interest, since it allows us to conclude that the nice properties of the minimum-backlog routing algorithm are maintained even when there are some mismatches between the current network state $\mathbf{X}(n)$ and the set of weights $\hat{\mathbf{X}}(n)$ used by the routing algorithm to route new requests. This mismatch can be due for example to a delay in distributing updated link cost vectors. We can thus state that finite delays in propagating the link state information may affect general performance indexes, such as the average transfer time, but they do not reduce the stability region of the routing algorithm. See also Section VIII.

We will not repeat the proof in case of wrong estimate of $\mathbf{X}$ for the more general cases considered in the next sections, but the same arguments hold also for those cases.

V. Results for Feasible Scheduling Policies

In the previous section we have considered an idealized scenario where links are always fully utilized whenever their virtual backlog is not null. This optimistic situation is not true in practice.

Thus, Equation (10) must be rewritten as:

$$\mathbf{X}(n+1) = \mathbf{X}(n) - \mathbf{W}(n) + \mathbf{R}^T(n)\mathbf{A}(n) \qquad (16)$$

where $\mathbf{W}(n)$ is a vector whose l-th element $w^l(n)$ represents the amount of work provided by link (queue) q^l to all the enqueued flows during time slot n, i.e., $\mathbf{W}(n)$ is the result of the *scheduling policy* implemented at the different links.

In general, $\mathbf{W}(n)$ is not fully specified by the unfinished work $\mathbf{X}(n)$, since it depends also on the real backlog that is (physically) enqueued at queues at time n. In the previous section we have assumed that all queues are physically backlogged whenever their virtual backlog is not null.

In this section, instead, we make the opposite assumption that the real backlog at queues along the flow paths can always be considered negligible. This can be obtained either by ideally adapting to network conditions the sending rate at the source, or similarly by enforcing that the amount of information permitted at the network ingress point does not exceed the service capacity along the path to the destination. We basically assume no physical backlog, and no need for storage capacity at queues. Being this the case, the same amount of work must be provided to a given flow by all the physical queues along its path, forcing an additional constraint on $\mathbf{W}$, which translates into a conservative approach. Under these assumptions, we introduce now the concept of *physically sustainable* (or just *sustainable*) work (or service rates) provided by queues to each flow.

Let $w^\pi(n) \geq 0$ be the amount of work provided by time n to the flows routed on path π by all the queues $l \in L(\pi)$, and $\mathbf{W}^\pi(n)$ the corresponding vector. The virtual backlog $x^\pi(n)$ induced at time n by flows routed on path $\pi \in \Pi(sd)$ on every queue $l \in L(\pi)$ must be identical. Let $\mathbf{X}^\pi(n)$ the vector of the $x^\pi(n)$.

Definition 3: The vector $\mathbf{W}(n)$ is physically sustainable if:

$$w^l(n) \;=\; \sum_{\pi\in\Pi(l)} w^\pi(n) \leq 1 \qquad \forall l \qquad (17)$$
$$w^\pi(n) \;\leq\; x^\pi(n) \qquad (18)$$

We call *feasible scheduling policy* a policy that at each time slot finds a sustainable $\mathbf{W}(n)$.

The scheduling policy computes either the vector $\mathbf{W}(n)$, i.e., the amount of provided work, or a set of guaranteed service rates, from which the provided work is derived according to some algorithm (e.g., using GPS [21]). We call[4] $b^\pi_{sd}(n)$ the minimum guaranteed service rate associated with the flow routed on path π at all queues, and $\mathbf{B}^\pi(n)$ the corresponding vector. The $b^\pi_{sd}(n)$ are constrained by a relation similar to (17):

$$\sum_{\pi\in\Pi(l)} b^\pi_{sd}(n) \leq 1 \qquad \forall l \qquad (19)$$

but do not need to satisfy the equivalent of (18), and do not necessarily depend from $\mathbf{X}(n)$. A simple way of deriving $w^\pi(n)$ from $b^\pi_{sd}(n)$ is the following:

$$w^\pi(n) = \min(x^\pi(n), b^\pi_{sd}(n)) \qquad (20)$$

In the sequel we refer to the correspondence (20) between $w^\pi(n)$ and $b^\pi_{sd}(n)$.

The dynamics of the path virtual backlogs are given by:

$$\mathbf{X}^\pi(n+1) = \mathbf{X}^\pi(n) - \mathbf{W}^\pi(n) + (\mathbf{R}^\pi(n))^T\mathbf{A}(n) \qquad (21)$$

We are now in a position to generalize the results presented in the previous section, proving that, under any admissible traffic pattern, a network adopting a routing similar to the minimum-backlogged-path routing in conjunction with a generic feasible scheduling policy is stable.

However, we first need to prove that traffic admissibility and traffic sustainability are equivalent also under the assumptions made in this section. The basic idea is that this equivalence holds if the service rates provided by the scheduling policy are matched with the average loads routed through network nodes. A GPS-like scheduler easily achieves this result.

A. Sustainability vs Admissibility

To prove that sustainability is equivalent to admissibility, we must show that there always exists a static routing function $\hat{\mathbf{R}}$ and a static (time invariant, so that we drop the dependence from n) feasible scheduling policy such that

$$b^\pi_{sd} > \hat{r}^\pi_{sd}\rho_{sd} \qquad \forall\pi \qquad (22)$$

[4] Subscripts sd are redundant in the notation, but kept for clarity and uniformity.

i.e., the service rate guaranteed to active flows routed on π by queues along π is greater than the average workload arrival rate on path π.

Lemma 1: Any *admissible* average arrival vector $E[\mathbf{A}]$ is *sustainable*.

Proof: According to the definition of admissibility, given an admissible traffic pattern, there exists a static routing algorithm $\hat{\mathbf{R}}$, such that, for some $\epsilon > 0$:

$$\hat{\mathbf{R}}^T E[\mathbf{A}] \leq \mathbb{1}(1-2\epsilon)$$

Let us assign to each path $\pi \in \Pi(sd)$ a static service rate $b^{\pi}_{sd} = (1+\epsilon)\hat{r}^{\pi}_{sd}\rho_{sd}$. Call $b^l = \sum_{\pi \in \Pi(l)} b^{\pi}_{sd}$ the link working rates, and $\mathbf{B}$ the corresponding vector.

By construction, $\mathbf{B} = (1+\epsilon)\hat{\mathbf{R}}^T E[\mathbf{A}] < \mathbb{1}$ defines a set of sustainable link working rates which stabilizes the network.

■

Note that, from the previous Lemma, it immediately follows that, for each admissible average arrival vector $\mathbf{A}$, $\hat{\mathbf{R}}^T E[\mathbf{A}]$ lies in the convex-hull of the sustainable service rates $\mathbf{B}$.

B. Main Results

We now extend the result of Theorem 2 to the general case of feasible scheduling policies, i.e., to the case where $\mathbf{W}(n)$ is constrained to be sustainable. Therefore, we need to define a scheduling algorithm that computes $\mathbf{W}(n)$ when the chosen dynamic routing algorithm is adopted.

Definition 4: We define the *max-scalar path scheduling policy* as the feasible scheduling policy computing the provided work $\mathbf{W}^{*\pi}(n)$ which maximizes the scalar product $\mathbf{W}^T\mathbf{X}^{\pi}(n)$:

$$\mathbf{W}^{*\pi}(n) = \arg\max_{\mathbf{W}} \mathbf{W}^T\mathbf{X}^{\pi}(n)$$

subject to (17), and (18).

We now define a modification of the minimum-backlogged-path routing (11).

Definition 5: A dynamic routing algorithm $\mathbf{R}^{*\pi}(n)$ such that

$$\mathbf{R}^{*\pi}(n) = \arg\min_{\mathbf{R}} \mathbf{A}^T(n)\mathbf{R}\mathbf{X}^{\pi}(n) \tag{23}$$

is called *minimum-backlogged-flow routing.*
This routing algorithm selects the path with the minimum virtual backlog, without accounting for the hop count along the path.

We introduce the following a preliminary result:

Lemma 2: Let

$$\mathbf{B}^{*\pi}(n) = \arg\max_{\mathbf{B}} \mathbf{B}^T\mathbf{X}^{\pi}(n)$$

subject to (19). It results:

$$[\mathbf{B}^{*\pi}(n) - \mathbf{W}^{*\pi}(n)]^T\mathbf{X}^{\pi}(n) \leq (\mathbf{B}^{*\pi}(n))^T\mathbf{B}^{*\pi}(n)$$

i.e, the difference between $(\mathbf{B}^{*\pi}(n))^T\mathbf{X}^{\pi}(n)$ and $(\mathbf{W}^{*\pi}(n))^T\mathbf{X}^{\pi}(n)$ is bounded by a finite constant.

Proof:

$$\begin{aligned}
&[\mathbf{B}^{*\pi}(n) - \mathbf{W}^{*\pi}(n)]^T\mathbf{X}^{\pi}(n) = \\
&\quad = (\mathbf{B}^{*\pi}(n))^T\mathbf{X}^{\pi}(n) - \max_{W} \mathbf{W}^T\mathbf{X}^{\pi}(n) \\
&\quad \leq (\mathbf{B}^{*\pi}(n))^T\mathbf{X}^{\pi}(n) - \min(\mathbf{X}^{\pi}(n), \mathbf{B}^{*\pi}(n))^T\mathbf{X}^{\pi}(n) \\
&\quad = \left[\mathbf{B}^{*\pi}(n) - \min(\mathbf{X}^{\pi}(n), \mathbf{B}^{*\pi}(n))\right]^T \mathbf{X}^{\pi}(n) \\
&\quad = \max(\mathbf{B}^{*\pi}(n) - \mathbf{X}^{\pi}(n), 0)^T\mathbf{X}^{\pi}(n) \\
&\quad \leq \max(\mathbf{B}^{*\pi}(n) - \mathbf{X}^{\pi}(n), 0)^T\mathbf{B}^{*\pi}(n) \\
&\quad \leq (\mathbf{B}^{*\pi}(n))^T\mathbf{B}^{*\pi}(n)
\end{aligned}$$

having chosen $\mathbf{W}^{\pi}$ according to (20) in the first inequality, and noted that $\max(a-b,0)b \leq \max(a-b,0)a$ in the second last inequality.

■

We are now ready to introduce our first main result.

Theorem 4: For any sustainable traffic pattern, a network is stable if the minimum-backlogged-flow routing and the max-scalar path scheduling policy are adopted.

Proof: We prove this result by applying the Lyapunov function methodology, following the same approach used in the proof of Theorem 2. Let us consider the quadratic Lyapunov function, as in Eq.(12). Stability is guaranteed if there exist $\epsilon > 0$ and $B > 0$ such that, for $||\mathbf{X}(n)|| > B$, the Lyapunov function drift is strictly negative:

$$\begin{aligned}
&\frac{E[\mathcal{L}(\mathbf{X}^{\pi}(n+1)) - \mathcal{L}(\mathbf{X}^{\pi}(n)) \mid \mathbf{X}^{\pi}(n)]}{||\mathbf{X}^{\pi}(n)||} = \\
&\quad = \frac{2E[\mathbf{A}^T(n)\mathbf{R}^{*\pi}(n)\mathbf{X}^{\pi}(n)] - 2(\mathbf{W}^{*\pi}(n))^T\mathbf{X}^{\pi}(n)}{||\mathbf{X}^{\pi}(n)||} \\
&\quad + \frac{o(||\mathbf{X}^{\pi}(n)||)}{||\mathbf{X}^{\pi}(n)||} \\
&\quad = \frac{2\left\{E[\mathbf{A}^T(n)\mathbf{R}^{*\pi}(n)] - (\mathbf{W}^{*\pi}(n))^T\right\}\mathbf{X}^{\pi}(n)}{||\mathbf{X}^{\pi}(n)||} + o(1) \\
&\quad < -\epsilon
\end{aligned}$$

having neglected at the numerator terms not comprising $\mathbf{X}^{\pi}(n)$.

To prove that the drift is negative, we define the routing algorithm $\mathbf{R}'^{\pi}(n) = \arg\min_{\mathbf{R}^{\pi}} E[\mathbf{A}^T(n)]\mathbf{R}^{\pi}\mathbf{X}^{\pi}(n)$; note that

$$\begin{aligned}
\mathbf{A}^T(n)\mathbf{R}^{*\pi}(n)\mathbf{X}^{\pi}(n) &= \min_{\mathbf{R}^{\pi}} \mathbf{A}^T(n)\mathbf{R}^{\pi}\mathbf{X}^{\pi}(n) \\
&\leq \mathbf{A}(n)\mathbf{R}'^{\pi}(n)\mathbf{X}^{\pi}(n)
\end{aligned}$$

thus

$$E[\mathbf{A}^T(n)\mathbf{R}^{*\pi}(n)]\mathbf{X}^{\pi}(n) \leq E[\mathbf{A}(n)]\mathbf{R}'^{\pi}(n)\mathbf{X}^{\pi}(n)$$

Finally:

$$\begin{aligned}
&E[\mathbf{A}(n)]^T\mathbf{R}'^{\pi}(n)\mathbf{X}^{\pi}(n) - (\mathbf{W}^{*\pi}(n))^T\mathbf{X}^{\pi}(n) = \\
&\quad = \min_{\mathbf{R}^{\pi}} E[\mathbf{A}^T(n)]\mathbf{R}^{\pi}\mathbf{X}^{\pi}(n) - (\mathbf{W}^{*\pi}(n))^T\mathbf{X}^{\pi}(n) \\
&\quad \leq E[\mathbf{A}^T(n)]\hat{\mathbf{R}}^{\pi}\mathbf{X}^{\pi}(n) - (\mathbf{W}^{*\pi}(n))^T\mathbf{X}^{\pi}(n) \\
&\quad = E[\mathbf{A}^T(n)]\hat{\mathbf{R}}^{\pi}\mathbf{X}^{\pi}(n) - \max_{\mathbf{W}} \mathbf{W}^T\mathbf{X}^{\pi}(n) \\
&\quad \leq E[\mathbf{A}^T(n)]\hat{\mathbf{R}}^{\pi}\mathbf{X}^{\pi}(n) - \max_{\mathbf{B}} \mathbf{B}^T\mathbf{X}^{\pi}(n) \\
&\qquad + o(||\mathbf{X}^{\pi}(n)||) < -\epsilon||\mathbf{X}^{\pi}(n)||
\end{aligned}$$

In the second-last equality the provided work $\mathbf{W}^\pi$ was approximated with the service rates $\mathbf{B}^\pi$ thanks to Lemma 2. Moreover, the last inequality holds since $E[\mathbf{A}^T(n)]\hat{\mathbf{R}}^\pi$ is in the convex hull of sustainable service rates $\mathbf{B}^\pi$ (due to Lemma 1) if $\mathbf{A}$ is admissible. ■

Note that we do not strictly need that $\mathbf{W}^*(n) = \arg\max_{\mathbf{W}} \mathbf{W}^T\mathbf{X}(n)$: we just need $\mathbf{W}^{*T}(n)\mathbf{X}(n) > E[\mathbf{A}^T(n)]\hat{\mathbf{R}}\mathbf{X}(n)$.

Unfortunately, the implementation of the max-scalar scheduling policy can be difficult, since it requires the solution of a difficult optimization problem (as defined in Def. 4) which is hard to implement in a distributed scenario.

We thus need to look for other pairs of dynamic routing algorithm and feasible scheduling policy which guarantee maximum throughput, while also being easier to implement in a distributed environment.

A possibly interesting feasible scheduling policy assigns bandwidth to flows according to:

$$b_{sd}^\pi(n) = \frac{x^\pi(n)}{\max_{l\in L(\pi)} \sum_{\pi'\in\Pi(l)} x^{\pi'}(n)} = \frac{x^\pi(n)}{\max_{l\in L(\pi)} x^l(n)} \tag{24}$$

We call this scheduling policy *max-backlog-proportional scheduling*; it provides service to flows proportionally to their virtual backlog with respect to the virtual backlogs of flows sharing the most congested link along the path. Note that it does not guarantee to fully utilize the link capacities, and that it does not require any knowledge of matrix $\boldsymbol{\rho}$. We also note that the properties shown in the sequel hold also when path virtual backlogs $x^\pi(n)$ are multiplied by finite constants (which may be useful to model the effective rates of TCP connections).

We now prove the existence of dynamic routing algorithms that, if used jointly with this scheduling policy, can stabilize the network under any admissible traffic pattern.

Theorem 5: If flows from $s \to d$ are routed on path π, with:

$$\pi = \arg\min_{\hat{\pi}\in\Pi(sd)} \max_{l\in L(\hat{\pi})} \sum_{\pi'\in\Pi(l)} x^{\pi'}(n) = \arg\min_{\hat{\pi}\in\Pi(sd)} \max_{l\in L(\hat{\pi})} x^l(n)$$

and the network adopts the max-backlog-proportional scheduling, then the network is stable under any admissible traffic. In the case of ties, i.e, when two paths π_1 and π_2 exist, such that $\max_{l\in L(\pi_1)} x^l(n) = \max_{l\in L(\pi_2)} x^l(n)$, the path with the least loaded second highest-load link is selected. In case of further tie, the third highest-load link is considered, and so on. In case links of π_1 present the same congestion on the sequence of highest loaded links of π_2, the shortest path will be selected. This routing algorithm will be called *minimum-congestion routing*, since flows $s \to d$ are routed along the path whose highest-loaded link is least loaded.

The proof is reported in Appendix . ■

VI. Flows with Exponentially Distributed Size

If flow lengths are exponentially distributed with parameter μ, the network can be modeled by a Markov Chain whose state descriptor $\mathbf{Z}^\pi(n)$ is an $|E|$-dimensional vector whose π-th element $z^\pi(n)$ represents the number of flows that are traversing path π at time n.

The dynamics of $z^\pi(n)$ are described by the following equation:

$$z^\pi(n+1) = z^\pi(n) + \sum_{sd} \gamma_{sd}(n) r_{sd}^\pi(n) - d^\pi(n)$$

where $\gamma_{sd}(n)$ represents the number of new flows from node s to node d arrived at the network during time slot n, and $d^\pi(n)$ is a random variable which denotes the number of flows that completed the data transfer at time slot n.

Rewriting the previous equation using vectorial notation, we obtain:

$$\mathbf{Z}^\pi(n+1) = \mathbf{Z}^\pi(n) + \mathbf{R}^\pi(n)^T\mathbf{\Gamma}(n) - \mathbf{D}^\pi(n) \tag{25}$$

being easy to verify that i) $E[\mathbf{\Gamma}(n)] = \mu E[\mathbf{A}(n)]$, and ii) $E[\mathbf{D}^\pi(n) \mid \mathbf{Z}^\pi(n)] = \mu \mathbf{W}^\pi(n)$. Thus:

$$E[\mathbf{Z}^\pi(n+1) - \mathbf{Z}^\pi(n) \mid \mathbf{Z}^\pi(n)] = \mu\mathbf{R}^\pi(n)^T E[\mathbf{A}(n)] - \mu\mathbf{W}^\pi(n)$$

which highlights the fact that the structure of (25) is identical to the structure of (21), under the variable substitution $\mathbf{X}^\pi \to \mathbf{Z}^\pi$. As a consequence, the results obtained in the previous section can be immediately extended for "companion" routing algorithms and scheduling policies which operate on $\mathbf{Z}^\pi$ rather than $\mathbf{X}^\pi$. A formal proof can be obtained by applying the stochastic Lyapunov functions obtained by substituting $\mathbf{Z}^\pi$ to $\mathbf{X}^\pi$.

The following results can be immediately derived.

We can re-define the dynamic minimum-backlogged-flow routing algorithm as the routing algorithm $\mathbf{R}^*(n)$ such that:

$$\mathbf{R}^{*\pi}(n) = \arg\min_{\mathbf{R}} \mathbf{A}^T(n)\mathbf{R}\mathbf{Z}^\pi(n)$$

We can re-define the max-scalar path scheduling policy as the policy $\mathbf{W}^{*\pi}(n)$ that maximizes the scalar product $\mathbf{W}^T\mathbf{Z}^\pi(n)$. That is:

$$\mathbf{W}^{*\pi}(n) = \arg\max_{\mathbf{W}} \mathbf{W}^T\mathbf{Z}^\pi(n)$$

subject to (17) and (18).

We can re-define the max-backlog-proportional scheduling policy to operate on the flow number $\mathbf{Z}^\pi$ rather than the path backlog $\mathbf{X}^\pi$, also explicitly considering different link capacities:

$$\begin{aligned} b_{sd}^\pi(n) &= \frac{z^\pi(n)/C^{l'}}{\max_{l\in L(\pi)} \frac{1}{C^l} \sum_{\pi'\in\Pi(l)} z^{\pi'}(n)} \\ &= \frac{z^\pi(n)/C^{l'}}{\max_{l\in L(\pi)} z^l(n)/C^l} \end{aligned}$$

where $C^{l'}$ is the capacity of the bottleneck link for path π.

We can finally re-define the minimum-congestion routing algorithm as follows:

$$\begin{aligned} \pi &= \arg\min_{\hat{\pi}\in\Pi(sd)} \max_{l\in L(\hat{\pi})} \frac{1}{C^l} \sum_{\pi'\in\Pi(l)} z^{\pi'}(n) \\ &= \arg\min_{\hat{\pi}\in\Pi(sd)} \max_{l\in L(\hat{\pi})} \frac{z^l(n)}{C^l} \end{aligned}$$

Then, we can state the following two theorems, which are direct consequences of Theorems 4, and 5.

Theorem 6: If flow lengths are exponentially distributed, for any sustainable traffic pattern, a network adopting the minimum-backlogged-flow routing algorithm and the max-scalar-path scheduling policy is stable.

Theorem 7: If flow lengths are exponentially distributed, for any sustainable traffic pattern, a network adopting the minimum-congestion routing algorithm and the max-backlog-proportional scheduling policy is stable.

This latter theorem refers to a routing algorithm that is quite similar to what is currently proposed for packet networks, since it corresponds to a minimum distance routing [9] adopting the L_∞ norm (i.e, where only the bottleneck link is considered).

VII. Flows with Generally Distributed Size

Since the assumption of exponential flow sizes may be unrealistic, we now discuss how the results obtained in the previous section for exponentially distributed flow sizes can be extended to flows with generally distributed size (with mean $1/\mu$ and finite polynomial moments).

First, we observe that in our model each link can be described by a single-server processor sharing queue if we assume that the flows uniformly share the available capacity (i.e., a max-backlog-proportional scheduling policy is adopted). Then we claim that the following property holds in our system:

$$\lim_{||\mathbf{Z}^\pi(n)||\to\infty} \frac{E[||\mu \mathbf{X}^\pi(n) - \mathbf{Z}^\pi(n)||]}{||\mathbf{Z}^\pi(n)||} = 0 \qquad (26)$$

i.e., the number of flows becomes a good estimator of the residual workload at the queue when the workload in the network becomes arbitrarily large. This property is known in the literature as *state space collapse* (or multiplicative state space collapse) [23], [24], and has been proved for several queuing systems, among which the G/G/1 Processor Sharing queue [25], [26]. Under the state space collapse assumption, we can extend the results obtained for exponentially distributed flows also to the generally distributed flow size, since i) routing algorithms and scheduling policies which operate on $\mathbf{Z}^\pi(n)$ and $\mathbf{X}^\pi(n)$ become equivalent when $\mathbf{Z}^\pi(n) \to \infty$; ii) policies operating on $\mathbf{X}^\pi(n)$ were already proved to maximize the network throughput.

VIII. Implementation Issues

After discussing the stability of dynamic routing algorithms, we now focus on the viability of an implementation based on current technology, and in particular in the context of IP networks. We refer to a single Autonomous System (AS), hence to IGP protocols. We concentrate on the minimum-congestion routing algorithm, because this seems to be the most interesting among the algorithms discussed in previous sections.

The algorithm works at the flow level, therefore being more suited to a virtual circuit technology, rather than a pure datagram network. We can identify four basic functionalities that must be supported by the network layer in order to implement the proposed algorithm:

- flow definition and identification
- route selection and pinning
- link cost measurement and distribution
- bandwidth sharing and scheduling

Some of these functionalities might be difficult to implement in a pure datagram network, and some solutions might be impractical. Hoverer, it is important to understand that almost all the dynamic QoS routing proposals face the same implementation difficulties in a pure datagram network. This is one of the reasons why they have not yet been widely adopted in the Internet.

A. Flow definition and identification

The definition and identification of new flows is required by all the algorithms we proposed. A flow may be any abstract aggregation of information, whose source and destination are the same, and that acts as an elastic source. A single data transfer is then a natural definition of flow, but also a set of data connections which have the same source and destination will work. While it is not possible to explicitly route all single data flows, the nature of Internet traffic, which comprises either quite long or very short connections, can be exploited to reduce the number of explicitly tracked flows, as proposed in [4]. Short connections can be interpreted as noise in the link state measurement (see Sect. VIII-C).

An effective way for the network to identify and classify flows is required, together with a mechanism to initiate the selection of a dynamic route for long-lived flows. Routers at the edge of the network can employ efficient flow classification algorithms [27] or hardware. Being the classification performed only at network edges, it does not pose critical scalability issues.

Note that sources should be elastic, i.e., capable to adapt their sending rate to the outcome of the scheduling policy, to guarantee negligible physical backlog in network nodes (see also Section VIII-D).

B. Route selection and pinning

When a new flow has been identified, its path π must be selected by the routing algorithm. This choice requires a complete knowledge of backlogs at all network nodes (see Section VIII-C). The path choice can be implemented directly at the source node, which can explicitly select the path, based on the knowledge of each link cost, and then signal the selection to all the nodes along the path. This approach does not require the flow identification at each node along the path.

A hop-by-hop route selection is also possible, in which each node independently selects the best next-hop when a new flow is identified. Indeed, routing decisions taken at an upstream node are not liable to be superseded by downstream nodes finding a locally-optimal alternative to the path chosen earlier by upstream nodes. Using the formalism provided in [1], it is immediate to verify that the minimum-congestion routing algorithm belongs to the class of algorithms that allow a consistent hop-by-hop implementation. As usual, proper mechanisms to avoid the selection of non simple paths (i.e., loops in the routing) must be adopted.

In both cases, a route pinning mechanisms is required, i.e., the path associated with a flow must remain the same until the flow ends. While route pinning is natural if explicit signaling and path selection is used, in the hop-by-hop implementation this could be achieved by explicitly adding an entry in the routing table when a new flow is identified along the path. When the flow ends, the explicit entry in the routing table can be removed.

C. Link cost measurement and distribution

The measurement of the link cost is at the base of every QoS routing algorithm. In our case, the metric to be measured is the current number of active flows. If explicit signaling is used when a new flow is routed, a simple counter is sufficient. Otherwise, in the hop-by-hop implementation, some inconsistencies on flow number may arise at different nodes.

Once the link cost has been measured by a node, it must be distributed to all other nodes in the network, if a distributed implementation of the routing algorithm is desired. Classic link state protocols, like OSPF or IS-IS, can be used to this purpose. Both the delay introduced by the distribution protocol, and the representation of metrics with finite precision introduce errors, which may affect the generic QoS algorithm performance, and cause inconsistencies in the path calculation. However, such information mismatch does not affect the stability properties of the proposed algorithms, if the error is bounded, as we proved in Theorem 2. From this point of view, all the algorithms presented are robust to measurement errors and distribution delays.

D. Bandwidth sharing and scheduling

The minimum-congestion routing algorithm maximizes the network stability region if link capacities are allocated according to the max-backlog-proportional scheduling policy. Any implementation of GPS-like schedulers is sufficient to allocate the proper bandwidth share to the flows traversing each link, provided that the information on the number (or the virtual backlog) of flows routed in the network is made available by the link state protocol. A GPS-like scheduler requires per-flow queuing at the different links.

If instead simple FIFO scheduling is implemented at network nodes, and flows share link resources according to the TCP congestion control mechanisms, we do not obtain scheduling policies similar to those considered in this paper. It is however possible to show that the properties proven above still hold when we modify our scheduling policies in order to always exploit the full link capacity on the bottleneck link of each path (this is note obtained by the max-backlog-proportional scheduling, as noted above), and to account for TCP unfairness to large-delay routes. This should lead to a scheduling policy that well accounts for the bandwidth sharing imposed by TCP. We plan to explore this possibility in future studies, and to try to extend our results to other types of scheduling policies.

We further note that in [28] several end-to-end congestion control protocols, different from TCP, have been studied, which approximate a number of fair bandwidth sharing criteria. These may be alternative approaches to implement the policies studied in this paper.

Given the previous considerations, and the flow-oriented nature of the algorithms, it seems more natural to favor a virtual circuit implementation, with respect to an hop-by-hop implementation, even if the latter is feasible. Therefore, explicit path mechanisms and signaling are required, and flow identification can either be triggered by users which explicitly inform the network of the new connection request, or be automatically performed by edge nodes. This scenario is compatible with the MPLS technology, in which Label Switched Paths are used to explicitly route flows, also fulfilling the route pinning requirement.

IX. Conclusions

In this paper we considered packet networks loaded by admissible traffic patterns, i.e. traffic patterns that, if optimally routed, do not overload network resources. We proved that several combinations of simple distributed dynamic routing algorithms and scheduling policies based upon link state information can achieve the same network throughput as optimal centralized routing and scheduling algorithms with complete traffic information.

Our proofs are based on abstract *flow-level* models of the network, consider *elastic* traffic, and are compatible with traffic engineering and QoS approaches being currently considered for IP networks.

We showed that maximum throughput is achieved even in case of temporary mismatches between the ideal link costs and those used by the routing algorithm. This implies that our proofs hold even for distributed implementations, which lead to errors in the estimation of routing metrics.

Special attention was given to the case of exponentially distributed flow sizes, which permit particularly simple routing and scheduling algorithms.

A discussion about implementation issues showed that the algorithms whose efficiency is proved in the paper are not unrealistic in the framework of the currently considered QoS approaches for the Internet, and for packet networks in general.

Acknowledgment

The authors would like to thank the anonimous reviewers for their preciuos comments. This work was supported by the Italian FIRB research project “Tango”.

References

[1] J.L.Sobrinho, “Algebra and Algorithms for QoS Path Computation and Hop-by-Hop Routing in the Internet,” *IEEE/ACM Transactions on Networking*, Vol. 10, N. 4, August 2002, pp. 541-550.

[2] J.L.Sobrinho, “Network Routing with Path Vector Protocols: Theory and Applications,” *ACM SIGCOMM 2003*, Karlsruhe, Germany, August 2003.

[3] G.Apostolopoulos, R.Guérin, S.Kamat, S.K.Tripathi, “Quality of Service Based Routing: A Performance Perspective,” *ACM SIGCOMM 1998*, Vancouver, Canada, September 1998.

[4] A.Shaikh, J.Rexford, K.Shin, “Load-Sensitive Routing of Long-lived IP Flows,” *ACM SIGCOMM 1999*, Cambridge, MA, August 1999.

[5] G.Apostolopoulos, D.Williams, S.Kamat, R.Guerin, A.Orda, T.Przygienda, “QoS Routing Mechanisms and OSPF Extensions,” *RFC 2676*, IETF, August 1999.

[6] Z.Wang, Y.Wang, L.Zhang, “Internet Traffic Engineering Without Full Mesh Overlaying,” *IEEE Infocom 2001*, Anchorage, AK, April 2001.

[7] A.Sridharan, R.Guérin, C.Diot, “Achieving Near-Optimal Traffic Engineering Solutions for Current OSPF/IS-IS Networks”, *IEEE Infocom 2003*, San Francisco, CA, March 2003.

[8] Z.Wang, J.Crowcroft, “QoS Routing for Supporting Multimedia Applications,” *IEEE JSAC*, Vol.14, N. 7, September 1996, pp 1228-1234.

[9] Q.Ma, P.Steenkiste, H.Zhang, “Routing High-Bandwidth Traffic in Max-Min Fair Share Networks,” *ACM SIGCOMM 1996*, Stanford, CA, August 1996.

[10] C.Casetti, R.Lo Cigno, M.Mellia, M.Munafò, Z.Zsoka, “A New Class of QoS Routing Strategies Based on Network Graph Reduction,” *IEEE Infocom 2002*, New York, NY, June 2002.

[11] A.Basu, A.Lin, S.Ramanathan, “Routing Using Potentials: A Dynamic Traffic-Aware Routing Algorithm,” *ACM SIGCOMM 2003*, Karlsruhe, Germany, August 2003.

[12] D.Berstekas, R.Gallager, *Data Networks,* Prentice Hall, 2nd Edition, 1987.

[13] L.Fratta, M.Gerla, L.Kleinrock, “The Flow Deviation Method - An Approach to the Store-and-Forward Communication Network Design,” *Networks*, Vol. 3, 1973, pp. 97-133.

[14] E.Rosen, A.Viswanathan, R.Callon, “Multi-Protocol Label Switching Architecture,” *RFC 3031*, IETF, January 2001.

[15] P.R.Kumar, S.P.Meyn, “Stability of Queueing Networks and Scheduling Policies”, *IEEE Transactions on Automatic Control*, Vol. 40, N.2, February 1995, pp. 251-260.

[16] S.P.Meyn, R.Tweedie, *Markov Chain and Stochastic Stability*, Springer-Verlag, 1993.

[17] L.Tassiulas, A.Ephremides, “Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks”, *IEEE Transactions on Automatic Control,* vol. 37, n. 12, December 1992, pp. 1936-1948.

[18] M.J.Neely, E.Modiano, C.E.Rohrs, “Dynamic Power Allocation and Routing for Time Varying Wireless Networks,” *IEEE Infocom 2003*, San Francisco, CA, April 2003.

[19] M.Armony, N.Bambos, “Queueing Dynamics and Maximal Throughput Scheduling in Switched Processing Systems”, *Queueing Systems*, Vol.44, 2003, pp.209-252.

[20] N.McKeown, A.Mekkittikul, V.Anantharam, J.Walrand, “Achieving 100% Throughput in an Input-Queued Switch,” *IEEE Transactions on Communications*, Vol. 47, N. 8, August 1999, pp. 1260-1272.

[21] A.K.Parekh, R.G.Gallager. “A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks - The Single Node Case,” *IEEE/ACM Transactions on Networking,* Vol. 13, June 1993, pp. 344-357.

[22] M.Bramson, “Instability of FIFO Queueing Networks,” *Annals of Applied Probability,* Vol. 4 (1994), pp. 414-431.

[23] R.J.Williams, ”An Invariance Principle for Semimartingale Reflecting Brownian Motions in an Orthant,” *Queueing Systems: Theory and Applications*, Vol. 30 (1998), pp. 5-25.

[24] M.Bramson, ”State Space Collapse with Application to Heavy Traffic Limits for Multiclass Queueing Networks”, *Queueing Systems: Theory and Applications*, Vol. 30 (1998), pp. 26-44.

[25] H.C.Gromoll, A.L.Puha, R.Williams, ”The Fluid Limit of a Heavily Loaded Processor Sharing Queue,” AMS 2000.

[26] H.C.Gromoll, ”Diffusion Approximation for a Processor Sharing Queue in Heavy Traffic,” *Annals of Applied Probability*, Vol.14 (2004), No.2, pp. 555-611.

[27] S.Singh, F.Baboescu, G.Varghese, J.Wang, “Packet Classification Using Multidimensional Cutting,” *ACM SIGCOMM 2003*, Karlsruhe, Germany, August 2003.

[28] J.Mo, J.Walrand “Fair End-to-End Window-based Congestion Control” *IEEE/ACM Transactions on Networking*, Vol. 8, N. 5, October 2000, pp. 556-567.

Appendix

Proof of Theorem 5

Let $x^l(n) = \sum_{\pi \in \Pi(l)} x^\pi(n)$. Given the state of the system $x^\pi(n)$, there exists a sufficiently large constant m_0 such that the minimum-congestion routing algorithm can be defined as, for $m > m_0$:

$$\mathbf{R}^*(n) = \arg\min_{\mathbf{R}} \sum_{sd} a_{sd}(n) \sum_{\pi \in \Pi(sd)} r^\pi_{sd} \sum_{l \in L(\pi)} \left(x^l(n)\right)^m \tag{27}$$

from which, clearly:

$$\sum_{sd} a_{sd}(n) \sum_{\pi \in \Pi(sd)} r^{*\pi}_{sd}(n) \sum_{l \in L(\pi)} \left(x^l(n)\right)^m \leq$$
$$\leq \sum_{sd} a_{sd}(n) \sum_{\pi \in \Pi(sd)} \hat{r}^\pi_{sd} \sum_{l \in L(\pi)} \left(x^l(n)\right)^m$$

reminding that $\hat{r}^\pi_{sd}$ defines a static routing algorithm which stabilizes the network. Note that, for any routing algorithm $\mathbf{R}$:

$$\sum_{sd} a_{sd}(n) \sum_{\pi \in \Pi(sd)} r^\pi_{sd}(n) \sum_{l \in L(\pi)} \left(x^l(n)\right)^m =$$
$$= \sum_{l} \sum_{\pi \in \Pi(l)} \sum_{sd} a_{sd}(n) r^\pi_{sd} \left(x^l(n)\right)^m$$

Thus:

$$\sum_{l} \sum_{\pi \in \Pi(l)} \sum_{sd} a_{sd}(n) r^{*\pi}_{sd} \left(x^l(n)\right)^m \leq$$
$$\leq \sum_{l} \sum_{\pi \in \Pi(l)} \sum_{sd} a_{sd}(n) \hat{r}^\pi_{sd} \left(x^l(n)\right)^m$$

Since the network traffic $\boldsymbol{\rho}$ is admissible, under $\hat{\mathbf{R}}$ the average load on every link must be strictly less then 1. Let u_M be the maximum link utilization under $\hat{\mathbf{R}}$. Thus:

$$\sum_{\pi \in \Pi(l)} \sum_{sd} \rho_{sd} \hat{r}^\pi_{sd} \leq u_M < 1 \qquad \forall l \in E$$

We prove the stability of the minimum-congestion algorithm defined by (27) by using the following Lyapunov function for an appropriate large m:

$$\mathcal{L}(\mathbf{X}) = \sum_{l} \frac{\left(x^l\right)^{m+1}}{m+1} = \sum_{l} \sum_{\pi \in \Pi(l)} x^\pi \frac{\left(x^l\right)^m}{m+1}$$

Since all the polynomial moments of flow sizes are finite, inequality (8) holds. As a consequence, we have just to show that the Lyapunov drift is negative when the queue size becomes large, i.e. that (9) also holds. Let us now compute the Lyapunov function drift. We define $\Delta x^l(n) = x^l(n+1) -$

$x^l(n)$:

$$\begin{aligned}
&\mathcal{L}(\mathbf{X}(n+1)) - \mathcal{L}(\mathbf{X}(n)) = \\
&\quad = \sum_l \frac{\left(x^l(n+1)\right)^{m+1}}{m+1} - \sum_l \frac{\left(x^l(n)\right)^{m+1}}{m+1} \\
&\quad = \sum_l \frac{\left(x^l(n+1)\right)^{m+1} - \left(x^l(n)\right)^{m+1}}{m+1} \\
&\quad = \sum_l \frac{\left(x^l(n) + \Delta x^l(n)\right)^{m+1} - \left(x^l(n)\right)^{m+1}}{m+1} \\
&\quad = \frac{1}{m+1} \sum_l \Big[\left(x^l(n)\right)^{m+1} + (m+1)(x^l(n))^m \Delta x^l(n) \\
&\qquad - \left(x^l(n)\right)^{m+1}\Big] + o(||\mathbf{X}(n)||^m) \\
&\quad = \sum_l (x^l(n))^m \Delta x^l(n) + o(||\mathbf{X}(n)||^m) \\
&\quad = \sum_l (x^l(n))^m \sum_{\pi\in\Pi(l)} [x^\pi(n+1) - x^\pi(n)] + o(||\mathbf{X}(n)||^m)
\end{aligned}$$

Noticing that $x^\pi(n+1) - x^\pi(n) = \sum_{sd} a_{sd}(n) r^{*\pi}_{sd}(n) - w^\pi(n)$, we get:

$$\begin{aligned}
&E[\mathcal{L}(\mathbf{X}(n+1)) - \mathcal{L}(\mathbf{X}(n)) \mid \mathbf{X}(n)] = \\
&\quad = \sum_l \sum_{\pi\in\Pi(l)} \left[\sum_{sd} \rho_{sd} r^{*\pi}_{sd}(n) - w^\pi(n)\right] \left(x^l(n)\right)^m \\
&\qquad + o(||\mathbf{X}(n)||^m)
\end{aligned}$$

Consider $\epsilon > 0$, and define the set $J^*_\epsilon(n)$ as the set of most congested links, or the set of links with a congestion level within ϵ from the most congested link:

$$J^*_\epsilon(n) = \left\{ l \in E \mid \frac{x^l(n)}{\max_{l'\in E} x^{l'}(n)} > 1 - \epsilon \right\} \tag{28}$$

Since the Lyapunov drift function is defined for large $||\mathbf{X}(n)||$, due to (20), $\mathbf{W}^\pi$ and $\mathbf{B}^\pi$ can be considered to be equivalent in $J^*_\epsilon(n)$. Hence, combining (24) and (28), we obtain:

$$\begin{aligned}
w^l(n) &= \sum_{\pi\in\Pi(l)} w^\pi(n) \\
&\approx b^l(n) = \sum_{\pi\in\Pi(l)} b^\pi(n) > 1 - \epsilon \qquad \forall l \in J^*_\epsilon(n)
\end{aligned}$$

In addition, for any large $B > 0$, there exists a m such that:

$$\left(\frac{x^l}{\max_{l'} x^{l'}}\right)^m < \frac{1}{B} \qquad \forall l \notin J^*_\epsilon \tag{29}$$

Thus:

$$\begin{aligned}
&\sum_l \sum_{\pi\in\Pi(l)} \left[\sum_{sd} \rho_{sd} r^{*\pi}_{sd}(n) - w^\pi(n)\right] \left(x^l(n)\right)^m = \\
&\quad = \sum_{l\in J^*_\epsilon} \sum_{\pi\in\Pi(l)} \left[\sum_{sd} \rho_{sd} r^{*\pi}_{sd}(n) - w^\pi(n)\right] \left(x^l(n)\right)^m \\
&\qquad + \sum_{l\notin J^*_\epsilon} \sum_{\pi\in\Pi(l)} \left[\sum_{sd} \rho_{sd} r^{*\pi}_{sd}(n) - w^\pi(n)\right] \left(x^l(n)\right)^m \\
&\quad < \sum_{l\in J^*_\epsilon} \left[\sum_{\pi\in\Pi(l)} \sum_{sd} \rho_{sd} \hat{r}^\pi_{sd} - (1-\epsilon)\right] \left(x^l(n)\right)^m \\
&\qquad + \sum_{l\notin J^*_\epsilon} \sum_{\pi\in\Pi(l)} \left[\sum_{sd} \rho_{sd} \hat{r}^\pi_{sd} - w^\pi(n)\right] \left(x^l(n)\right)^m \\
&\quad \le \sum_{l\in J^*_\epsilon} (u_M - 1 + \epsilon) \left(x^l(n)\right)^m \\
&\qquad + \sum_{l\notin J^*_\epsilon} (u_M - w^l(n)) \left(x^l(n)\right)^m \\
&\quad < \sum_{l\in J^*_\epsilon} (u_M - 1 + \epsilon) \left(x^l(n)\right)^m + u_M \sum_{l\notin J^*_\epsilon} \left(x^l(n)\right)^m \\
&\quad < -\sum_{l\in J^*_\epsilon} 2\epsilon \left(x^l(n)\right)^m + u_M \sum_{l\notin J^*_\epsilon} \left(x^l(n)\right)^m
\end{aligned}$$

where the last inequality holds for any choice of ϵ such that $3\epsilon < 1 - u_M$. We note that the second term can be made strictly smaller in module than the first term for sufficiently large B in (29). Indeed, by choosing $B > \frac{|E|}{\epsilon}$, it results:

$$\sum_{l\notin J^*_\epsilon} \left(x^l(n)\right)^m < \epsilon \max_l (x^l(n))^m < \epsilon \sum_{l\in J^*_\epsilon} \left(x^l(n)\right)^m$$

■