Modeling the Interactions of Congestion Control and Switch Scheduling

Alex Shpiner and Isaac Keslassy
Department of Electrical Engineering
Technion - Israel Institute of Technology
Haifa, 32000, Israel
{shalex@tx, isaac@ee}.technion.ac.il

***Abstract*—In this paper, we study the interactions of user-based congestion control algorithms and router-based switch scheduling algorithms. We show that switch scheduling algorithms that were designed without taking into account these interactions can exhibit a completely different behavior when interacting with feedback-based Internet traffic. Previous papers neglected or mitigated these interactions, and typically found that flow rates reach a fair equilibrium. On the contrary, we show that these interactions can lead to extreme unfairness with temporary flow starvation, as well as to large rate oscillations. For instance, we prove that this is the case for the MWM switch scheduling algorithm, even with a single router output and basic TCP flows. We also show that the iSLIP switch scheduling algorithm achieves fairness among ports, instead of fairness among flows. Finally, we fully characterize the network dynamics for both these switch scheduling algorithms.**

I. Introduction

A. Congestion Control vs. Switch Scheduling

This paper is about combining two conflicting parallel views of the Internet: a *user-centric view*, which considers that user-based end-to-end congestion control algorithms regulate the Internet and that routers are just passive elements of the Internet; and a *router-centric view*, which considers that router-based switch scheduling algorithms regulate the Internet and that users are just passive elements of the Internet.

Both the congestion control and the switch scheduling algorithms have the same common goal: *regulate Internet traffic* so as to maximize link utilization, minimize packet loss, and provide fairness among flows. However, they use quite different means. User-based congestion control algorithms like TCP regulate traffic by decreasing the rates of flows that experience losses, and increasing the rates of flows that do not. On the other hand, router-based switch scheduling algorithms like Maximum Weight Matching (MWM) regulate traffic by providing more services to long backlogged queues, and less services to small queues.

While both traffic regulation algorithms reach high performance when considered independently, we will show that their interacting actions might conflict when put together, and eventually cause more harm than good.

Figure 1 illustrates these issues on a toy model consisting of two flows queued at two different inputs and destined for the same output. Assume that these are TCP flows of rates λ_1 and λ_2, and that the switch implements MWM by always servicing the flow with the longest queue. Independently, both traffic regulation algorithms work fine: if the flows were using TCP but not MWM, e.g. by sharing the same FIFO queue, then they would reach the well-known TCP rate equilibrium [1]. Likewise, if the two flows were using MWM but not TCP, by using a non-adaptive algorithm with fixed flow rates, then they would both receive 100% throughput as long as $\lambda_1 + \lambda_2 < 1$ [2].

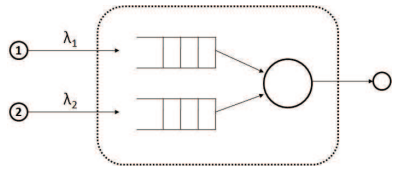


Fig. 1. Simple network of two flows with common output link.

The problem arises when the two traffic regulation algorithms interact. If the queue of the first flow gets larger, MWM will keep servicing it, and therefore the first flow will increase its rate even further in a vicious circle, because TCP will keep receiving ACKs. On the other hand, the second flow will not receive services and get starved. Thus the first flow will overtake all the network resources. The combination of the congestion control and the switch scheduling will cause an *extreme unfairness*, which was absent when they were each alone.

For router designers, this is no trivial result. It might mean that their carefully-designed switch-scheduling algorithms, which work perfectly with all the benchmarks based on non-responsive flows, might break down when introduced in real Internet networks with responsive TCP flows.

For network researchers, this is no trivial result either. It might change the perceived value of many well-known results. For instance, the Birkhoff-von Neumann (BvN) switch scheduling algorithm, which measures the average flow arrival rates and can provide proportional service rates, is known to be fair for non-responsive flows [3]. In fact, it is one of the only switch scheduling algorithms that are known to provide both throughput and fairness guarantees in practical switch architectures. However, as in the example above, providing more services to a responsive flow might increase its arrival rate in turn, thus increasing its share of the total traffic and leading again to a vicious circle with extreme unfairness. Therefore, it

might be that the BvN scheduling algorithm simply does not fit real Internet traffic, with the vast majority of the bandwidth consisting of TCP responsive flows [4], [5].

These considerations show that congestion control and switch scheduling algorithms *cannot* be designed and analyzed without taking into account their interactions, both in practical router benchmarks and in theoretical network models.

Further, to make things even worse, the example above could also lead to different results, depending on the network topology. For instance, if the queue of the first flow is the longest one and keeps getting serviced, its service rate might exceed its arrival rate, and therefore its size will decrease, until both queue sizes are equal and the second flow gets serviced as well. So it might be that the queue sizes get equalized and stay equal. Or it might also be that the two flows alternately overtake the whole link capacity. Unfortunately, as seen in this paper, *all* these behaviors are possible, and highly depend on network parameters. Therefore, this example also illustrates the inherent *analysis complexity* associated to the interactions between congestion control and switch scheduling.

B. Related Work

Known models of congestion control algorithms often assume *output-queued switching*, i.e. the existence of a single passive queue shared by all the flows destined to a switch-output bottleneck link. For instance, these models have dealt with flow rate equilibria [1], router buffer sizing [6], TCP dynamics [7], TCP fairness [1], Weighted Fair Queueing (WFQ) [8], and Active Queue Management (AQM) analysis [9]. Unfortunately, output-queued switching cannot be implemented in high-speed routers because of its required memory speedup [10]. Therefore, we will analyze the more realistic input-queued routers and their associated switch scheduling algorithms.

Known models of switch scheduling algorithms often assume *non-responsive traffic* to analyze algorithms like MWM [2], BvN [3], and the heuristic iSLIP [11]. These models attempt to achieve more realistic conditions by using admissible non-responsive flows with either variable-size packets [12], fixed traces [13], router measurements [14], or networked switches [15]. But most of Internet traffic is actually responsive. In this paper, we also consider responsive flows such as TCP flows.

Recently, research works have started modeling the interactions of responsive flows with switch scheduling algorithms. First, [16], [17] model the interaction of TCP sources and the MWM scheduling algorithm. Their model relies on the RED AQM scheme, and they convincingly prove that there always exists a fair system equilibrium point. However, RED mitigates the effects of MWM in that it discriminates against longer queues, while MWM favors them. As a consequence, this model does not reflect the possible extreme unfairness and large rate oscillations that can occur without AQM.

In addition, [18], [19] measure packet delays in a real router fed with a closed-loop ns2-generated TCP traffic. Such an approach can accurately reflect delays at real Internet routers. However, it is dependent on the router implementation, and cannot model arbitrary switch scheduling algorithms.

Further, [20], [21] model the interactions of responsive flows with switch scheduling algorithms in wireless networks. However, they assume congestion control policies that are fundamentally different from TCP.

C. Contributions

In this paper, we attempt to provide a first characterization of the interactions between congestion control and switch scheduling algorithms, using mostly TCP flows and droptail queues. We compare the performances of an output-queued switch; an input-queued switch implementing iSLIP, the scheduling algorithm on which the Cisco 12000 GSR router is based [11]; and an input-queued switch implementing MWM.

By restricting our model to the tractable single-port case, we characterize the system equilibria when they exist, and compare their fairness properties. For instance, we show that output-queued switches are *fair for flows*, while iSLIP-based input-queued switches are *fair for ports*. We also characterize the cases in which MWM leads to *extreme unfairness* with temporary flow starvation.

Further, we discover three different modes of MWM: *starvation, oscillation and equalization*. We find that these modes have fundamentally different properties, and highly depend on the topology parameters.

Last, we completely describe the *network dynamics* for both the iSLIP and MWM scheduling algorithms, using a set of differential equations. We show that iSLIP can be modeled by considering each (input, output) queue as a full output-queued switch. We also find that the behavior of MWM is based on synchronized congestion cycles.

The rest of the work is organized as follows. After defining our model in Section II, we successively analyze the fairness of OQ, iSLIP-based IQ and MWM-based IQ switches under TCP traffic in Sections III, IV and V. Then, we characterize the network dynamics of iSLIP and MWM in Section VI. We finally show simulation results for these models in Section VII.

Due to space limits, proofs are omitted and can be found in [22].

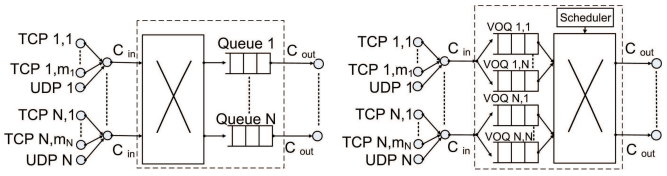
II. Model and Notations

We now introduce and define our model and notations. We first describe the general network topology and the congestion control of each flow, and then focus on the switch and on its scheduling algorithm.

A. Network Model

Figures 2(a) and 2(b) illustrate the general network topology, using a central switch that can be either output-queued or input-queued.

Network — The network includes N groups of flow sources. Each group $1 \leq i \leq N$ consists of m_i persistent TCP-Reno sources and several UDP (or more generally non-responsive) sources modeled as a single UDP Poisson source. All these flow sources are connected to a group aggregation switch, which is



(a) Output-queued switch. (b) Input-queued switch.

Fig. 2. Network topology based on switch.

connected with its own link of capacity C_{in} to input port i of the $N \times N$ switch. The switch is then connected to the flow destinations with links of capacity C_{out}. Therefore, packets sent by the sources are routed through the group aggregation switch to the main switch, and then to their destination. Acknowledgements (ACKs) come back in the same way. We now make a few simplifying assumptions on the network properties to make the problem more tractable. First, we assume that for these flows, the only bottleneck links are the forwarding links from the switch to the flow destinations.

Assumption 1: The only queues in the network are the packet queues in the switch. In particular, all link capacities but C_{out} are assumed to be infinite, and the backward propagation times are assumed to be fixed.

Round Trip Time —There are $m = \sum_{i=1}^{N} m_i$ TCP flows. For each TCP flow k, let $(w^k(t), Q^k(t), C^k(t))$ respectively denote the congestion window size, number of queued packets, and switch service rate of flow k at time t. Also, let $RTT^k(t)$ and τ^k denote its total (respectively propagation) round-trip time (RTT), i.e. the total time from source to destination and backwards with (without) counting queueing time.

In this paper, we will neglect sub-RTT variations of time-dependent rates, in order to avoid intractable delayed non-linear differential equations. For instance, if $Q^k(t)$ packets of flow k are currently queued and they are currently serviced at rate $C^k(t)$, then we assume that an entering packet from flow k will stay in the queue for $Q^k(t)/C^k(t)$ time-slots. Therefore, the total round-trip time is

$$RTT^k(t) = \tau^k + \frac{Q^k(t)}{C^k(t)} \tag{1}$$

Further, for each input i and output j, let $\mathcal{S}_{ij}$ be the set of TCP flows going through input i and output j. Then, for simplicity, we will assume that all flows in $\mathcal{S}_{ij}$ have the same propagation time.

Assumption 2: The propagation RTT of all flows $k \in \mathcal{S}_{ij}$ is equal and denoted $\tau_{ij}(t) \stackrel{\triangle}{=} \tau^k(t)$.

We now want to characterize the number of packets of each flow in the network. We first make a simplifying assumption to avoid distinguishing between services and departures.

Assumption 3: The service rate $C^k(t)$ of flow k always equals its departure rate, i.e. if $C^k(t) > 0$ then there are always packets from flow k to service in the queue, so $Q^k(t) > 0$.

Window — The total congestion window size $W_{ij}(t)$ of TCP flows in $\mathcal{S}_{ij}$ is denoted $W_{ij}(t) \stackrel{\triangle}{=} \sum_{k \in S_{ij}} w^k(t)$. Let $\tilde{w}^k(t)$ denote the number of packets in the network from flow k at time t, including ACKs. Then, since packets depart from the queue at rate $C^k(t)$ and take a round-trip propagation time of τ^k to come back, there are $C^k(t) \cdot \tau^k$ packets on the links, in addition to the $Q^k(t)$ packets in the queue, hence

$$\tilde{w}^k(t) = C^k(t) \cdot \tau^k + Q^k(t) \tag{2}$$

Moreover, by definition of the congestion window, assuming that TCP does not use the delayed-ACKs feature, we can model [6]

$$w^k(t) \approx \tilde{w}^k(t), \tag{3}$$

which is usually accurate unless flow k just experienced a congestion, in which case $w^k(t)$ falls faster than $\tilde{w}^k(t)$.

B. Switch Model

We now define the notations used for the switch arrivals, schedules, and services.

Arrivals — For each (input, output) pair (i, j), we denote $\lambda_{ij}(t)$ the total rate of packets arriving at input i and destined for output j. We decompose this arrival traffic into two types:

- *TCP traffic*, with arrival rate $\lambda_{ij}^k(t)$ for each flow $k \in \mathcal{S}_{ij}$, yielding a total arrival rate $\lambda_{ij}^{TCP}(t)$; and
- *Poisson UDP traffic*, with fixed total arrival rate λ_{ij}^{UDP}.

Thus, we have: $\lambda_{ij}(t) = \lambda_{ij}^{TCP}(t) + \lambda_{ij}^{UDP} = \sum_{k \in \mathcal{S}_{ij}} \lambda_{ij}^k(t) + \lambda_{ij}^{UDP}$.

Queues — We will assume that time is slotted, and that all data packets have a fixed size, such that each switch output can serve exactly one packet per time-slot.

Let $Q_{ij}(t)$ denote the number of packets arrived at input i, destined to output j, and queued in the switch at time t. We will respectively denote the number of queued TCP and UDP packets as $Q_{ij}^{TCP}(t)$ and $Q_{ij}^{UDP}(t)$. We saw that the number of queued packets from flow $k \in \mathcal{S}_{ij}$ is $Q^k(t) \stackrel{\triangle}{=} Q_{ij}^k(t)$. Therefore: $Q_{ij}(t) = Q_{ij}^{TCP}(t) + Q_{ij}^{UDP}(t) = \sum_{k \in \mathcal{S}_{ij}} Q_{ij}^k(t) + Q_{ij}^{UDP}(t)$. Likewise, the number of queued packets arrived at input i (respectively destined to output j) is $Q_{i\cdot}(t) = \sum_{j=1}^{N} Q_{ij}(t)$ (respectively $Q_{\cdot j}(t) = \sum_{i=1}^{N} Q_{ij}(t)$).

Services — We saw that TCP flow $k \in \mathcal{S}_{ij}$ receives a service rate of $C^k(t) \stackrel{\triangle}{=} C_{ij}^k(t)$. Likewise, the total service rate of all flows belonging to the (input, output) pair (i, j) is denoted $C_{ij}(t)$, including $C_{ij}^{TCP}(t)$ for TCP flows and $C_{ij}^{UDP}(t)$ for UDP flows, so that $C_{ij}(t) = C_{ij}^{TCP}(t) + C_{ij}^{UDP}(t) = \sum_{k \in \mathcal{S}_{ij}} C_{ij}^k(t) + C_{ij}^{UDP}(t)$.

C. Switch Architecture

We will distinguish two types of switches. First, an $N \times N$ *output-queued (OQ)* switch (Figure 2(a)) contains N queues, located at the output ports of the switch. As packets arrive, they are transferred immediately to their corresponding output queue j of length $Q_{\cdot j}(t)$.

An $N \times N$ *input-queued (IQ)* switch (Figure 2(b)) is built using N buffers, located at the input ports of the switch. Each input buffer i is shared dynamically between N *virtual output queues (VOQs)*, which correspond to the N outputs and have

total length $Q_{i\cdot}(t)$. When a packet arrives at input i and is destined to output j, it is stored in the corresponding VOQ, denoted VOQ_{ij}, of length $Q_{ij}(t)$.

In an IQ switch, after packet arrivals, a centralized switch scheduler decides on a match between the N input ports and the N output ports, so that no input (resp. output) is matched to more than one output (resp. input). Then, the scheduler picks the head-of-line packets of the selected VOQs to depart according to this match. The scheduler can follow any switch scheduling algorithm in order to decide which packet to serve. Scheduling algorithms considered in this paper include:

- *iSLIP*, a round-robin-based algorithm [11]. In iSLIP, each input (output) keeps a pointer to its preferred output (input), which rotates in a round-robin order once it is matched. Using an iterative process, the scheduler attempts to find a match by giving preference to the inputs and outputs indicated in the pointers. Note that iSLIP reduces to a simple *round-robin (RR)* scheduler on a vector of N VOQs, e.g. when there is only one input or output with active flows.
- *Maximum Weight Matching (MWM)*, which maximizes the weight of the match, with weights given by the queue lengths [2]. Intuitively, MWM favors larger VOQs. Note that MWM reduces to the *Longest Queue First (LQF)* policy on a vector of N VOQs.

In both switch architectures, the total buffer size at the switch is NB, i.e. B per output in the OQ switch and B per input in IQ switch. Further, all buffers implement a *droptail* policy, i.e. an arriving packet is dropped iff its buffer is full. We will define the set of congestion times for flow k by $\mathcal{T}^k$, where $t \in \mathcal{T}^k$ iff the size Q of the queue that contains flow k satisfies $Q(t^-) < B$ and $Q(t) = B$.

D. Single-Port Model

To get a better grasp of the problem, we will introduce and consider the single output-port model, in which a single output has active flows. Thus, the switch reduces to an $N \times 1$ switch, with simpler notations and switch scheduling algorithms. In this case, we will simplify notations by defining $\lambda_i \triangleq \lambda_{i,1}$, $Q_i \triangleq Q_{i,1}$, and so on. Further, as mentioned above, the iSLIP and MWM switch scheduling algorithms respectively reduce to the round-robin and LQF algorithms.

III. Fairness of OQ switches

In the next sections, we want to compare OQ and IQ switches from the point of view of fairness. To do so, we first define a simple fairness measure. Then, when considering the single-output case, we show that OQ switches are *fair*.

A. Fairness Measures

Our objective is to analyze the fairness of OQ and IQ switches, i.e. the way in which the available output link capacity is divided between flows. We first define Jain's fairness index [23], and then apply it to compare the performance of the switches.

Definition 1 (Jain's Fairness): Jain's fairness index for m flows is

$$F \triangleq \frac{\left(\sum_{i=1}^{m} C_i\right)^2}{m \cdot \sum_{i=1}^{m} C_i^2} \tag{4}$$

Since we assume a FIFO droptail queueing policy, it is hard to analyze the precise behavior of each flow. Therefore, we make a simplifying assumption on flows sharing the same queue.

Assumption 4: Two flows sharing the same queue have equal dropping probabilities. Further, their service rates are proportional to their queue sizes.

B. Fairness analysis

We will now analyze the fairness measure of OQ switches, and later compare it with IQ switches. For simplicity, we consider the *single output-port case*, in which all flows are switched to the same output port j. In this fairness analysis, we assume that all round-trip times are equal, and that there is no UDP traffic. We rely on the following approximation of the steady-state throughput of a TCP flow k with round-trip time RTT^k [1]:

$$C^k = \frac{\sqrt{2}}{RTT^k \cdot \sqrt{d^k}} \tag{5}$$

where C^k and d^k are the steady-state average values of the capacity $C^k(t)$ and the dropping rate $d^k(t)$. We neglect the difference between the average over time and the average seen by packet arrivals. The next theorem shows that the throughput of all flows in the output-queued switch is divided equally at the output link. (We remind that all proofs are presented in [22].)

Theorem 1 (OQ Switch Throughput): In the OQ switch defined above, the throughput of flow k is:

$$C^k = \frac{C_{out}}{\sum_{i=1}^{N} m_i} \tag{6}$$

Example 1: Consider a 2×1 OQ switch with 10 flows in the first input ($m_1 = 10$) and a single flow in the second input ($m_2 = 1$). Then, the service rate of each flow is

$$C^k = \frac{C_{out}}{m_1 + m_2} = \frac{C_{out}}{11}, \tag{7}$$

independently of its input.

Theorem 2: The OQ switch *maximizes Jain's fairness index*, achieving $F = 1$.

IV. Fairness of IQ Switches with iSLIP Scheduling

We saw above that OQ switches are fair. We now want to analyze iSLIP-based IQ switches. We prove that IQ switches using iSLIP scheduling are *unfair* in the general case, and show that they provide *port-fairness* instead of *flow-fairness*.

In order to analyze the fairness of iSLIP-based IQ switches, we assume the same setting as in the analysis of OQ fairness, with a single output-port. In such a setting, the iSLIP algorithm reduces to a simple round-robin (RR) scheduling scheme. In the next theorem, we show that each input i receives an equal

share of the output capacity, divided equally among its $m_i > 0$ flows.

Theorem 3 (iSLIP Throughput): In an IQ switch with iSLIP scheduling, the throughput of flow k in input i is

$$C_i^k = \frac{C_{out}}{N \cdot m_i} \tag{8}$$

Example 2: Consider again the network from Example 1, this time with an IQ switch using an iSLIP scheduler. Then, the service rate of each flow in the first input port is $C^k = C_{out}/20$, and the service rate of the flow in the second input port is $C^k = C_{out}/2$. This is clearly an *unfair* allocation among flows.

Based on the Cauchy-Schwarz inequalities, the following theorem shows that iSLIP is unfair under Jain's fairness.

Theorem 4: The iSLIP-based IQ switch is *unfair* by Jain's fairness criteria, unless all inputs have exactly the same number of flows. In particular, its Jain's fairness index is

$$F = \frac{N^2}{\left(\sum_{i=1}^{N} m_i\right) \cdot \left(\sum_{i=1}^{N} \frac{1}{m_i}\right)}. \tag{9}$$

V. Fairness of IQ Switches with MWM Scheduling

A. Starvation Mode vs. Oscillation Mode

We now analyze the fairness of IQ switches with MWM scheduling. For simplicity, we want to analyze the 2×1 case mentioned in the Introduction and shown in Figure 1. In such a case, the MWM algorithm is reduced to a simple LQF algorithm. We first neglect timeouts and UDP traffic, and later take them into account.

There are two conflicting intuitions on the expected results in the 2×1 case. First, we might believe that once a queue becomes large, the MWM scheduler keeps servicing it, and so its congestion window will keep growing until the flow takes control over the whole service rate and causes other flows to temporarily starve. So the MWM scheduler might be extremely *unfair* in such a *starvation mode*.

On the other hand, if a flow has the largest queue and keeps getting serviced, its queue can empty out faster, and then another flow will in turn have a larger queue and get service, thus overcoming the first flow. The service rate of each flow will oscillate between 0 and the full capacity C_{out}. So over a long average, the MWM scheduler might actually be somehow more *fair* in this *oscillation mode*.

The following analysis shows that both intuitions can be correct, and both the *starvation* and the *oscillation modes* can occur, depending on the network parameters. For instance, let's assume that at some time t_0 the first queue is longer than the second one:

$$Q_1(t_0) > Q_2(t_0) \tag{10}$$

Then the *starvation mode* occurs when this strict inequality keeps holding at all times $t \geq t_0$, both in *stable phases* (when queue sizes keep growing) and in *congestion phases* (when queue sizes fall).

Figure 3 illustrates the typical behavior of the *starvation mode*, in which the first flow keeps prevailing and the second

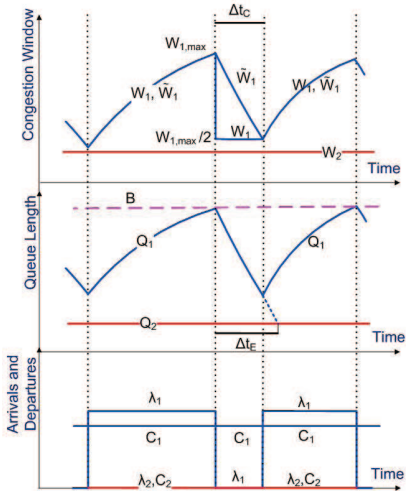


Fig. 3. Starvation mode for two TCP flows in a 2×1 MWM switch

flow is starved. Since we always have $Q_1(t) > Q_2(t)$, the first flow keeps getting serviced at rate $C_1 = C_{out}$. Therefore it keeps increasing its window size, and its corresponding queue arrival, until $Q_1 = B$. This causes a packet drop, and the window size is halved.

There is then a *race condition* between Δt_C, the time before both the window and the queue of the first flow start growing again, and Δt_E, the time it takes to equalize the queues lengths Q_1 and Q_2. As we will prove, when $\Delta t_C < \Delta t_E$, the first queue is always longer than the second one, and therefore the network stays in *starvation mode*. However, if $\Delta t_C > \Delta t_E$, the two queue lengths get equal, and the other flow might start growing faster, thus the network enters an *oscillation mode*.

In other words, during stable phases, a single prevailing queue is always being serviced, and the other queue is starved — but in *starvation mode*, the same queue is always prevailing, while in *oscillation mode*, the identity of the prevailing queue might change during the congestion phase. As stated in the following theorem, the mode depends in fact on the network topology.

Theorem 5: In the MWM-based IQ scheduler described above, assume that $Q_1(t_0) > Q_2(t_0)$ at time t_0. Then the switch is in *starvation mode* with $Q_1(t) > Q_2(t)$ for all $t \geq t_0$ iff the buffer size B satisfies

$$B > C_{out} \cdot \tau_1 + 2Q_2(t_0) \tag{11}$$

Furthermore, if $B \leq C_{out} \cdot \min(\tau_1, \tau_2)$, the switch is always in *oscillation mode*.

In particular, if $Q_2(t_0) = 0$, then the condition for the *starvation mode* corresponds to the well-known rule-of-thumb for the buffer size of an OQ switch [6]. With such a buffer size, we guarantee that the buffer of the first flow never goes empty, and therefore that it is always picked by the MWM scheduler.

Note that when we assume the existence of timeouts in starvation mode, Q_2 slightly grows at each timeout. However, even through the second queue is serviced from time to time,

the fundamental network properties are unchanged and it is still in *starvation mode*, with a negligible service rate for Q_2.

B. UDP Flows and Equalization Mode

We now want to analyze the influence of UDP flows on the network. We show that when the UDP flows have a low rate, their influence is negligible and we still have the same *starvation* and *oscillation modes*. However, for a slightly higher UDP flow rate, we prove the apparition of a third mode, the *equalization mode*, which keeps all queue sizes equal.

Assume that $Q_1(t_0) > Q_2(t_0)$ at time t_0. The intuition is that starvation will happen whenever $\frac{dQ_1}{dt}(t_0) > \frac{dQ_2}{dt}(t_0)$, i.e. $Q_1(t_0)$ is longer than $Q_2(t_0)$ and their difference keeps increasing. Otherwise, if $\frac{dQ_1}{dt}(t_0) < \frac{dQ_2}{dt}(t_0)$ and their difference keeps decreasing, queue 2 will exceed queue 1 at some time t_1 (i.e., $Q_2(t_1) > Q_1(t_1)$), and queue 2 will be serviced in turn. Therefore, we may obtain in turn $\frac{dQ_1}{dt}(t_1) > \frac{dQ_2}{dt}(t_1)$, and eventually queue 1 will exceed again queue 2. Thus, no queue will always prevail. Further, if this equalization happens fast, both queue sizes will remain nearly equal.

We first make the following simplifying assumption, and deduce the conditions for this *equalization mode*.

Assumption 5 (Arrivals and departures of UDP packets): We assume that the rate of the UDP packets is sufficiently low relatively to the service rate, so that during congestions the amount of dropped UDP packets is negligible. Therefore $C_{ij}^{UDP}(t) = \lambda_{ij}^{UDP}$.

Theorem 6 (Equalization mode): In the MWM-based IQ scheduler described above, the switch is in *equalization mode* at time t_0 whenever the arrival rate of UDP packets λ^{UDP} is sufficiently large and satisfies

$$\lambda_2^{UDP} > \frac{C_{out}}{Q(t_0) + C_{out} \cdot \tau_1} \text{ and } \lambda_1^{UDP} > \frac{C_{out}}{Q(t_0) + C_{out} \cdot \tau_2}.$$

In particular, if

$$\lambda_2^{UDP} > \frac{C_{out}}{B + C_{out} \cdot \tau_1} \text{ and } \lambda_1^{UDP} > \frac{C_{out}}{B + C_{out} \cdot \tau_2},$$

then the switch is always in *equalization mode*. Further, if UDP traffic is negligible and satisfies

$$\lambda_2^{UDP} < \frac{C_{out}}{B + C_{out} \cdot \tau_1} \text{ and } \lambda_1^{UDP} < \frac{C_{out}}{B + C_{out} \cdot \tau_2},$$

then as previously the switch is either in a finite-time *starvation mode* or in *oscillation mode*.

C. Fairness measures of MWM switch modes

We now analyze the fairness of the *starvation*, *oscillation*, and *equalization modes* in the simple 2×1 switch example shown in Figure 1, where each input queue serves a single flow.

Starvation Mode — In this case one of the queues is always being serviced, while the other is always starved, i.e. $C_1 = C_{out}, C_2 = 0$. We establish the following fairness result showing that the starvation mode is fundamentally unfair.

Theorem 7: In starvation mode, Jain's index is $F = \frac{1}{2}$.

Oscillation Mode — Assume that in oscillation mode, the flow prevailing is determined at each congestion in a round-robin manner. Then we obtain:

Theorem 8: Denote $\alpha_i = \tau_i \cdot C_{out} + B$. Then in oscillation mode, Jain's fairness index is $F = \frac{(\alpha_1^2+\alpha_2^2)^2}{2(\alpha_1^4+\alpha_2^4)}$.

Equalization Mode — We get the following theorem:

Theorem 9: In equalization mode, Jain's fairness index is $F = \frac{(\tau_1+\tau_2)^2}{2(\tau_1^2+\tau_2^2)}$.

VI. Network Dynamics using IQ Switches

In the next section we introduce more general models that rely on differential equations to model the network dynamics, while not restricting the number of inputs and the number of flows per input.

A. Model

We consider again the simplified *single-output* switch model. We now want to describe the network dynamics in the cases of the iSLIP and MWM switch scheduling algorithms. To do so, we first use *many small building blocks*, which describe the behaviors of the network components. Then, we connect them in a single set of equations. Finally, we reduce this set of general equations to a *simplified set of equations*, from which all other equations can be deduced. For instance, the simplified set only considers queue sizes and services rates — and once we solve it, we can deduce window sizes, arrival rates, long-term rate averages, fairness measures, etc.

We will see that the simplified set of equations has an interesting structure: it is always a *double set of equations*, reflecting the two sides of the interactions, i.e. both the *congestion control* and the *switch scheduling* algorithms. In fact, there are *two equations per flow*, one corresponding to the congestion control and one to the switch scheduling. In total, there are $2(m + N)$ equations, for the m TCP and N UDP flows. Further, the congestion control equations are different when TCP is in *stable phase* and *congestion phase*, i.e. between drops and during drops.

There is still one step left beyond this double set of equations. We need to determine when a flow has a packet drop and enters congestion. For instance, we previously defined $\mathcal{T}^k$, the set of congestion times for flow k. Likewise, we define the set of congestion times for input i by $\mathcal{T}_i$, where $t \in \mathcal{T}_i$ iff $Q_i(t^-) < B$ and $Q_i(t) = B$. Then if input i experiences congestion, not necessarily all flows going through this input will experience congestion as well — only those that experience packet drops. Further, a flow with more packets has more chance to experience packet drops. Thus, we need a model linking queue congestion and flow congestion. We will simply use the mean-field model from [7].

In the remainder, we describe the two simplified sets of equations for iSLIP and MWM. We remind that the full proofs are presented in [22].

B. Network Dynamics Theorems

First, we present the dynamics of the iSLIP-based network topology. In the next theorem, the switch-scheduling equations

are based on the intuition that iSLIP equally divides output capacity among incoming ports, and divides a port capacity among flows proportionally to their number of queued packets (Equation (12)). In addition, the congestion-control equations successively model TCP flows in stable phase, TCP flows in congestion phase, and UDP traffic (Equation (13)).

Theorem 10 (iSLIP Dynamics): The dynamics of Internet traffic going through an iSLIP switch can be modeled using the following set of $2(m+N)$ equations on the $2(m+N)$ flow variables $\{(Q^k(t), C^k(t))_{1\le k\le m}, (Q_i^{UDP}(t), C_i^{UDP}(t))_{1\le i\le N}\}$:
(i) $m+N$ switch scheduling equations:

$$\begin{cases} C^k(t) = \frac{Q^k(t)}{\sum_{k'\in\mathcal{S}_i} Q^{k'}(t)+Q_i^{UDP}(t)} \cdot \frac{C_{out}}{N} & \forall i, k \in \mathcal{S}_i \\ C_i^{UDP}(t) = \frac{Q_i^{UDP}(t)}{\sum_{k'\in\mathcal{S}_i} Q^{k'}(t)+Q_i^{UDP}(t)} \cdot \frac{C_{out}}{N} & \forall i \end{cases} \tag{12}$$

(ii) $m+N$ congestion control equations, reflecting stable phases and congestion phases:

$$\begin{cases} \frac{d}{dt}(Q^k(t)+C^k(t)\tau^k)^2 = 2C^k(t) & \text{if } t \notin \mathcal{T}^k \\ Q^k(t^+)+C^k(t^+)\tau^k = \frac{Q^k(t^-)+C^k(t^-)\tau^k}{2} & \text{if } t \in \mathcal{T}^k \\ \frac{dQ_i^{UDP}}{dt} = \lambda_i^{UDP} - C_i^{UDP}(t) \end{cases} \tag{13}$$

The next theorem presents the dynamics of the MWM-based network topology. The switch-scheduling equations express the full service rate provided to the longest queue by MWM (Equation (15)). As in Theorem 10, the congestion-control equations model TCP and UDP flows (Equation (16)).

Theorem 11 (MWM Dynamics): The dynamics of Internet traffic going through an MWM switch can be modeled using the following set of $2(N+m)$ equations on the $2(N+m)$ input variables $\{(Q^k(t), C^k(t))_{1\le k\le m}, (Q_i^{UDP}(t), C_i^{UDP}(t))_{1\le i\le N}\}$:
(i) $m+N$ switch scheduling equations: let $\mathcal{A}(t)$ denote the set of inputs with the longest queue at time t, i.e.

$$\mathcal{A}(t) = \{i : Q_i = \max_j Q_j\}, \tag{14}$$

then

$$\begin{cases} C_i^k(t) = \frac{\sum_{k'\in\mathcal{S}_i} C^{k'}(t)+C_i^{UDP}(t)}{\sum_{k'\in\mathcal{S}_i} Q^{k'}(t)+Q_i^{UDP}(t)} \cdot Q_i^k(t) \\ C_i^{UDP}(t) = \frac{\sum_{k'\in\mathcal{S}_i} C^{k'}(t)+C_i^{UDP}(t)}{\sum_{k'\in\mathcal{S}_i} Q^{k'}(t)+Q_i^{UDP}(t)} \cdot Q_i^{UDP}(t) \\ \sum_{k\in\mathcal{S}_i} Q_i(t) = \sum_{k\in\mathcal{S}_j} Q_j(t) \quad \text{if} \quad i,j \in \mathcal{A}(t) \\ \frac{d}{dt}\sum_{k\in\mathcal{S}_i} Q_i^k(t) = 0 \quad \text{if} \quad i \notin \mathcal{A}(t) \\ \sum_{k=1}^m C^k + \sum_{i=1}^N \lambda_i^{UDP} = C_{out} \end{cases} \tag{15}$$

where the number of independent equations for each equation line is successively $(m-N, N, |\mathcal{A}(t)|-1, N-|\mathcal{A}(t)|, 1)$, yielding a total of $m+N$.
(ii) $m+N$ congestion control equations, reflecting stable phases and congestion phases:

$$\begin{cases} \frac{d}{dt}(Q^k(t)+C^k(t)\tau^k)^2 = 2C^k(t) & \text{if } t \notin \mathcal{T}^k \\ Q^k(t^+)+C^k(t^+)\tau^k = \frac{Q^k(t^-)+C^k(t^-)\tau^k}{2} & \text{if } t \in \mathcal{T}^k \\ \frac{dQ_i^{UDP}}{dt} = \lambda_i^{UDP} - C_i^{UDP}(t) \end{cases} \tag{16}$$

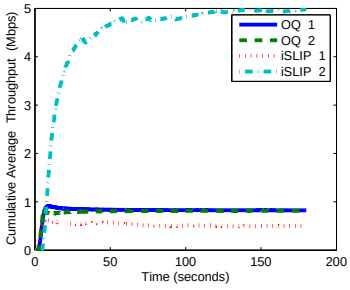


Fig. 4. Simulation of OQ and IQ-iSLIP cumulative average throughput.

As mentioned above, the full proofs appear in [22]. We also explain there why the model for the $N \times 1$ iSLIP switch is in fact a combination of N independent models of 1×1 switches, i.e. N FIFO queues.

VII. Simulations

A. Simulation Settings

We now want to evaluate the correctness of our models by comparing them with simulation results. We ran ns2 simulations of the network dynamics, and compared them with Matlab implementations of the differential equations in the iSLIP- and MWM-based IQ switch models.

In our simulations, we used default ns2 protocol implementations. For $i, j \ge 0$, we assumed that the round-trip propagation time of flows at input i and output j is $\tau_{ij} = (i+j+1)\cdot\tau_{00}$, with a base propagation time $\tau_{00} = 100$ ms. We also assumed a uniform packet size of 1 KB.

B. Fairness of OQ and IQ-iSLIP Switches

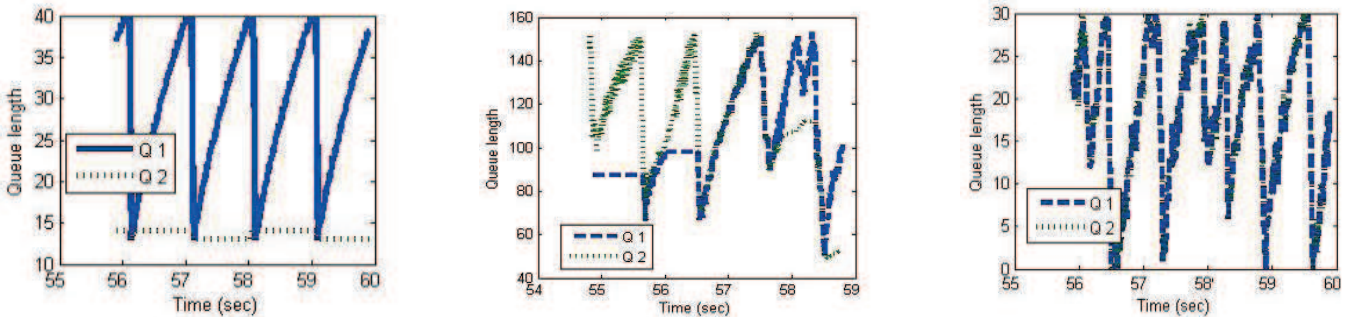
Figure 4 displays simulation results for the 2×1 switch example with 11 flows, as discussed in Examples 1 and 2. It plots the cumulative average throughput of one flow from each input, assuming both OQ and iSLIP-based IQ. The simulation used $C_{out} = 10$ Mbps, $B = 28$ KB, and a total average rate of UDP flows equal to 1% of the output link capacity C_{out}.

The figure confirms the results presented in the analysis. In the OQ switch, the throughput of different flows equalizes over time even if they are from different inputs, thus resulting in a *fair allocation*. However, in the IQ-iSLIP switch, the throughput of the flow from the second input tends to be ten times larger than the throughput of each of the ten flows from the first input, thus resulting in a *large unfairness*.

C. MWM Modes

Figures 5(a), 5(c) and 5(b) show the evolution of the instantaneous queue lengths of each input in the three MWM modes, assuming the 2×1 switch setting. All these figures were obtained using the same switch architecture, but different network topology conditions (different buffer sizes, propagation times, and output capacity).

Figure 5(a) shows the *starvation mode*, where queue 1 is the prevailing serviced queue and queue 2 is the starved queue. It



(a) 2×1 MWM – Starvation mode. (b) 2×1 MWM – Oscillation mode. (c) 2×1 MWM – Equalization mode.

Fig. 5. Fairness simulation graphs for all modes.

used a single flow per input, no UDP packets, $C_{out} = 1$ Mbps and $B = 41$ KB.

Figure 5(b) shows the *oscillation mode*, where only one of the queues gets full service rate at each time. In between two full-service states the queue apparently goes through an equalization phase between $t = 565$ seconds and $t = 575$ seconds. As opposed to the *starvation mode*, we can see that the full service is passing from one queue to another. It used five flows per input, no UDP packets, $C_{out} = 5$ Mbps and $B = 150$ KB.

Finally, Figure 5(c) plots the *equalization mode*, in which queue lengths are kept equal. It used a single flow per input, $C_{out} = 2$ Mbps, a total UDP rate of $20\% \cdot C_{out}$, and $B = 31$ KB.

These simulations can be used to validate Theorems 5 and 6. For instance, in the starvation mode settings, the following condition of Theorem 5 holds:

$$C_{out} \cdot \tau_1 + 2Q_2(t_0) \approx 2 \cdot 10^6/8 \cdot 0.1 + 2 \cdot 14 \cdot 10^3/8 = 40.5\text{KB} < 41\text{KB} = B$$

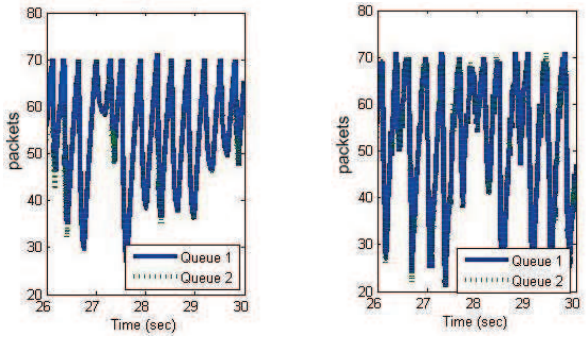
D. MWM Dynamics

Figures 6(a) and 6(b) compare the MWM switch dynamics in an ns2-based network simulation and in an implementation of the differential-equations model, both being run *under the same topology conditions*. We assumed a 2×1 switch five TCP flows per input, using $C_{out} = 5$ Mbps, a total UDP rate of $5\% \cdot C_{out}$, and $B = 70$ KB.

In both plots, the two queues appear to be in *equalization mode*, with both queue plots barely distinguishable. The queue dynamics seem quite similar in the model and in the simulation, thus providing a partial validation of the model. In particular, there is similarity in the minimal values, maximal values, and slopes of the respective functions. Incidentally, we present additional simulation results in [22] that confirm the model of iSLIP switch dynamics as well.

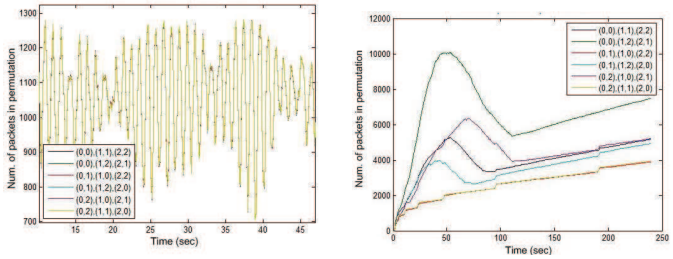
E. MWM Modes in $N \times N$ Switches

We finally want to evaluate the behavior of MWM in $N \times N$ switches. Such switches are much harder to analyze than $N \times 1$ switches, because of the many interactions between queues. We show below that their observed behavior in simulations reflects the MWM modes analyzed in $N \times 1$ switches.



(a) Model. (b) Simulation.

Fig. 6. 2×1 MWM switch dynamics with five TCP flows per input.



(a) Equalization mode. (b) Starvation mode.

Fig. 7. Simulation graphs of 3x3 MWM switch with 100 flows per VOQ.

Figures 7(a) and 7(b) illustrate the behavior of a 3×3 MWM switch under different topology parameters. Both plot the $3! = 6$ possible permutations weights, i.e. the total number of packets in the corresponding VOQs. Both assume 100 flows per (input,output) pair, i.e. a total of 900 flows. Further, Figure 7(a) used $C_{out} = 100$ Mbps, $B = 2.5$ MB, and $\tau_{00} = 100$ ms, while Figure 7(b) used $C_{out} = 10$ Mbps, $B = 10$ MB, and $\tau_{00} = 1$ ms.

We can see that in Figure 7(a), the switch is in *equalization mode*, under which all permutation weights tend to stay equal. On the other hand, in Figure 7(b), the switch is in *starvation mode*, with a single permutation having a weight higher than the others, and therefore always being served. (Note that other permutation weights steadily increase because of UDP and

timeout packets that keep arriving.) Therefore, the $N \times N$ switch dynamics reflect the dynamics analyzed in the $N \times 1$ switch; instead of dealing with packets queued in a specific queue, these are now the dynamics of all packets queued in a specific permutation.

VIII. Discussions

Let's now briefly discuss the correctness and generality of the assumptions made in this paper.

Single bottleneck — Assumption 1 presumes a single bottleneck in the network, and therefore neglects the influence of the other queues. This assumption relies on the observation that in the Internet, few flows practically have more than one bottleneck, and they mostly depend on their most congested queue [6]. Thus, this assumption seems realistic enough. However, we also assumed that the congestion only affects packets, not ACKs. This assumption is too restrictive, and ACK congestion is left for future study.

Equal round trip times — Assumption 2 neglects the RTT variations between flows in the same input port. In simulations, we varied the RTTs of different flows, while keeping them ordered by their RTTs to have similar RTTs in each port. We only found an impact of 1–2 % on the resulting flow capacities.

Non-empty queues — Assumption 3 relies on non-empty queues in the iSLIP switch. While this assumption is obviously wrong in the general case, we found that it mostly holds when buffer sizes are large enough. For instance, in our simulations, queue were typically empty less than 1 % of the time.

Drop-tail queues — Assumption 4 presupposes equal dropping probabilities for flows at the same queue. In simulations, we found that this assumption held when averaged over some sufficiently large time period (over 5 seconds), as long as the number of flows was large enough and the loss rate was reasonable.

UDP loss rate — Assumption 5 neglects the number of lost UDP packets compared to the total number of lost packets. We found that it held in simulations as well, as long as the total UDP rate was negligible.

IX. Conclusions

In this paper we modeled the interactions of user-based congestion control algorithms and router-based switch scheduling algorithms. Using single-port switches, we found that these interactions can lead to extreme unfairness and flow starvation, as well as to large rate oscillations. Further, we discovered three modes of MWM behavior, namely the starvation, oscillation and equalization modes. We also modeled the dynamics of both iSLIP and MWM switches, and showed in simulation results that our models were quite close to simulated dynamics.

In this paper, *none* of the studied arbitration modes in IQ switch schemes was found to be fair, further emphasizing the fairness issues resulting from the interactions of congestion control and switch scheduling.

In this paper, we did not exhibit any fair and efficient IQ scheme. Given our assumptions, iSLIP can be seen as less unfair than MWM, because it arbitrates equally across ports and does not discriminate against flows with large RTTs. However, iSLIP does not always provide 100% throughput [11]. Finding a fair scheme that guarantees 100% throughput is not an easy task — we conjecture that it can be reached using credit-based fairness mechanisms, but leave it for future work.

Acknowledgement

This work was partly supported by European Research Council Starting Grant n° 210389.

References

[1] F. Kelly, "Mathematical modelling of the Internet," In *Mathematics Unlimited - 2001 and Beyond*, Springer-Verlag, 2001.

[2] N. McKeown, V. Anantharan, and J. Walrand, "Achieving 100% throughput in an input-queued switch," *IEEE Infocom '96*, vol. 1, pp. 296–302, San Francisco, CA, March 1996.

[3] C. S. Chang, W. J. Chen, and H. Y. Huang, "On service guarantees for input buffered crossbar switches," *IEEE IWQoS'99*, pp. 79–86, London, UK, 1999.

[4] C. Fraleigh *et al.*, "Packet-level traffic measurements from the Sprint IP backbone," *IEEE Network*, vol. 17, pp. 6–16, 2003.

[5] M. Fomenkov *et al.*, "Longitudinal study of Internet traffic in 1998-2003," *WISICT*, vol. 58, pp. 1–6, 2004.

[6] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," *A*CM SIGCOMM, Portland, OR, 2004.

[7] M. Wang, "Mean-field analysis of buffer sizing," *Globecom'07*, Washington DC, Nov. 2007.

[8] H. Hassan, O. Brun, J. M. Garcia, and D. Gauchard, "Integration of streaming and elastic traffic: a fixed point approach," *SIMUTools*, 2008.

[9] T. Bu and D. F. Towsley, "A fixed point approximation of TCP behavior in a network," *ACM Sigmetrics*, 2001.

[10] F. Abel *et al.*, "Design issues in next-generation merchant switch fabrics," *IEEE/ACM Transactions on Networking*, vol. 15, no. 6, pp. 1603–1615, 2007.

[11] N. McKeown, "The iSLIP scheduling algorithm for input-queued switches," *IEEE Transactions on Networking*, vol. 7, no. 2, pp. 188–201, Apr. 1999.

[12] M. Ajmone Marsan, *et al.*, "Packet-mode scheduling in input-queued cell-based switches," *IEEE/ACM Transactions on Networking*, vol. 10, no. 5, pp. 666–678, Oct. 2002.

[13] P. Giaccone, M. Mellia, L. Muscariello, and D. Rossi, "Switches under real internet traffic", *IEEE HPSR*, Phoenix, Arizona, USA, April 2004.

[14] N. Hohn *et al.*, "Bridging router performance and queuing theory," *ACM Sigmetrics*, New York, June 2004.

[15] M. Andrews and L. Zhang, "Achieving stability in networks of input-queued switches," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 848–857, 2003.

[16] P. Giaccone, E. Leonardi and F. Neri, "On the behavior of optimal scheduling algorithms under TCP sources," *International Zurich Seminar on Communications*, pp. 94–97, Feb. 2006.

[17] P. Giaccone, E. Leonardi and F. Neri, "On the interaction between TCP-like sources and throughput-efficient scheduling policies," *Technical report*, Politecnico di Torino, July 2006.

[18] R. Chertov, S. Fahmy, and N. B. Shroff, "A black-box router profiler," *IEEE Global Internet*, May 2007.

[19] R. Chertov, S. Fahmy, and N. B. Shroff, "A device-independent router model, *Infocom 2008*, Phoenix, Arizona, May 2008.

[20] M.J. Neely, E. Modiano and C.-P. Li "Fairness and optimal stochastic control for heterogeneous networks, *Infocom '05*, pp. 1723–1734, Miami, FL, Mar. 2005.

[21] A. Eryilmaz and R. Srikant, "Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control, *Infocom '05*, pp. 1794–1803, Miami, FL, Mar. 2005.

[22] A. Shpiner and I. Keslassy, "Modeling the interaction of congestion control and switch scheduling," *Technical Report TR08-03*, Comnet, Technion, Israel. Available on http://www.ee.technion.ac.il/~isaac/papers.html

[23] R. Jain, D. M. Chiu, and W. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared systems," *DEC Research Report TR-301*, 1984.