Modeling the Interactions of Congestion Control and Switch Scheduling

Alexander Shpiner*, Isaac Keslassy

Department of Electrical Engineering, Technion - Israel Institute of Technology, Haifa, 32000, Israel

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.

Keywords: TCP, Congestion Control, Switch Scheduling.

1. Introduction

1.1. 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.

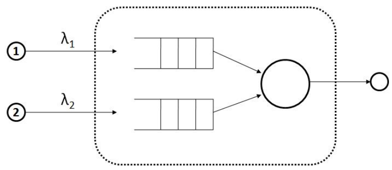


Figure 1: Simple network of two flows with common output link.

In this work we compare the scheduling schemes of Input-queued (IQ) switches and Output-queued (OQ) switches. The OQ switch is used as a simplified ideal switch model, while the IQ switch represents a more realistic model [11, 27, 28, 29, 38]. The main difference between them is the location of the queues. The OQ switch architecture uses queues after the switching fabric, but needs very high speed to write packets to the buffers. On the other hand, the IQ switch model removes the need for a high buffer-writing speed by using queues before the switching fabric, but in exchange limits the switching possibilities, and adds the complexity of switch scheduling schemes [28]. Note that in this paper, all queues are assumed to be drop-tail.

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 is using MWM by always servicing the flow with the longest queue. Independently, both traffic regulation algorithms (TCP and MWM) would have worked 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 [24, 31, 37]. Likewise, if the two flows

*Corresponding author (phone: 972-48295738, fax: 972-48295757).
This paper was presented in part at IWQoS ’09, Charleston SC, July 2009. Additions to the conference version include new theorems and results, complete proofs that were previously omitted for space reasons, and additional simulation results, as detailed in the cover letter.
This work was partly supported by European Research Council Starting Grant No. 210389.
Email addresses: shalex@tx.technion.ac.il (Alexander Shpiner), isaac@ee.technion.ac.il (Isaac Keslassy)

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 $\lambda_1 + \lambda_2 < 1$ [11, 27, 38].

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, one of the most significant results in switch-scheduling theory is that MWM provides 100% throughput for all non-responsive flows with admissible rates. But when over 90% of the traffic consists of responsive TCP traffic, the value of this result needs to be put in perspective, as shown in the toy model. Likewise, 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 [8, 26, 40]. 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 [15, 16].

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 served, 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 served 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 work, *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.

1.2. 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 [24, 31, 37], router buffer sizing [5, 32, 35, 41], TCP dynamics [36, 39], TCP fairness [24, 25, 37], Weighted Fair Queueing (WFQ) [21], and Active Queue Management (AQM) analysis [7, 12]. Unfortunately, output-queued switching cannot be implemented in high-speed routers because of its required memory speedup [1]. Therefore, it is not possible to rely on the implementation of a WFQ or AQM mechanism in a single output queue, without considering input queues as well. On the contrary, 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 [11, 27, 38], BvN [8, 26, 40], and the heuristic iSLIP [29, 28]. These models attempt to achieve more realistic conditions by using admissible non-responsive flows with either variable-size packets [6, 17, 2], fixed traces [20], router measurements [22], or networked switches [4, 3]. But most of Internet traffic is actually responsive. In this work, 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, [18, 19] model the interaction of TCP sources and the MWM scheduling algorithm. The work focuses on the dynamic interaction in packet networks between regulated Additive-Increase Multiplicative-Decrease (AIMD) traffic sources and max-scalar scheduling policies at switches. The latter was proved to be optimal in terms of throughput for stationary unregulated traffic sources. The average dynamics of AIMD traffic sources and switch queues are described through a system of delay differential equations. The studies find that AIMD sources and max-scalar switches co-exist well. AIMD fluid equations of congestion window size and queue size are presented. Their model relies on the RED AQM scheme, and the studies 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, [9, 10] 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, [13, 30] 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.

1.3. 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 [29]; and an input-queued switch implementing MWM.

We restricted our model to the tractable single output-port case, equivalent to a *server* model, in which the iSLIP-based input-queued switch becomes a round-robin (RR) server, and the MWM-based input-queued switch becomes a longest-queue-first (LQF) server. Using this model, we characterize the system equilibria when they exist, and compare their fairness properties. For instance, we show that output-queued servers are *fair for flows*, while RR servers are *fair for ports*. We also characterize the cases in which LQF leads to *extreme unfairness* with temporary flow starvation.

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

Then, we completely describe the *network dynamics* for both the RR and LQF scheduling algorithms in this server model, using a set of differential equations. We show that RR can be modeled by considering each (input, output) queue as a full output-queued switch. We also find that the behavior of LQF is based on synchronized congestion cycles, during which queues have an equal size, and grow and fall together.

Last, we provide some intuition on the behavior of iSLIP and MWM in a full switch using simulations. We show that iSLIP typically keeps its port-fairness properties, while under certain conditions, MWM tends to equalize the sum of queue sizes in VOQ (virtual output queue) permutations.

The rest of the work is organized as follows. After defining our model in Section 2, we successively analyze the fairness of OQ, RR-based IQ and LQF-based IQ switches under TCP traffic in the server model in Sections 3, 4 and 5. Then, we characterize the network dynamics of RR and LQF in the server model in Section 6. We finally show simulation results for these models in Section 7.

2. 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.

2.1. Network Model

Figures 2 and 3 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 connected with its own link of capacity C_{in} to input port i of the

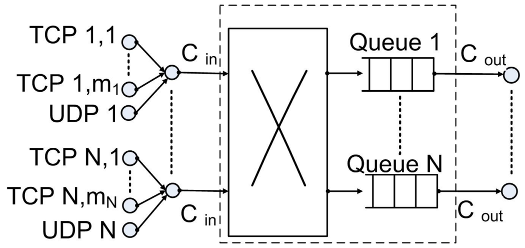


Figure 2: Network topology based on an output-queued switch.

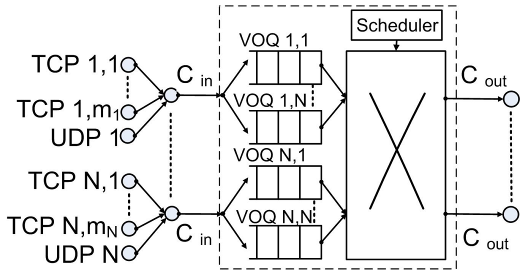


Figure 3: Network topology based on an input-queued switch.

$N \times N$ 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 $w^k(t)$ denote the congestion window size, $Q^k(t)$ denote the number of queued packets and $C^k(t)$ denote the switch service rate of flow k at time t. Also, let $RTT^k(t)$ denote the total round-trip time (RTT), i.e. the total time from source to destination and backwards including queueing time, and τ^k denote the propagation round-trip time (RTT), i.e. the total time from source to destination and backwards excluding queueing time.

In this work, 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 served 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 $\tau_{ij}(t) \triangleq \tau^k(t)$.*

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. The next assumption on non-empty queues is needed to build a theoretical model, even though it does not necessarily hold in practice. It is further discussed in Section 8.

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 $W_{ij}(t) \triangleq \sum_{k\in\mathcal{S}_{ij}} w^k(t)$. 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 [5]

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

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

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

Thus, we have:

$$\begin{aligned} \lambda_{ij}(t) &= \lambda^{TCP}_{ij}(t) + \lambda^{UDP}_{ij} \\ &= \sum_{k\in\mathcal{S}_{ij}} \lambda^k_{ij}(t) + \lambda^{UDP}_{ij}. \end{aligned} \tag{4}$$

Queues — We will assume that all data packets have a fixed size. 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 denote the number of queued TCP packets as $Q^{TCP}_{ij}(t)$ and UDP packets as $Q^{UDP}_{ij}(t)$. We saw that the number of queued packets from flow $k \in \mathcal{S}_{ij}$ is $Q^k(t) \triangleq Q^k_{ij}(t)$. Therefore:

$$\begin{aligned} Q_{ij}(t) &= Q^{TCP}_{ij}(t) + Q^{UDP}_{ij}(t) \\ &= \sum_{k\in\mathcal{S}_{ij}} Q^k_{ij}(t) + Q^{UDP}_{ij}(t). \end{aligned} \tag{5}$$

Likewise, the number of queued packets arrived at input i is $Q_{i\cdot}(t) = \sum_{j=1}^{N} Q_{ij}(t)$ and the number of queued packets destined to output j is $Q_{\cdot j}(t) = \sum_{i=1}^{N} Q_{ij}(t)$.

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

$$\begin{aligned} C_{ij}(t) &= C^{TCP}_{ij}(t) + C^{UDP}_{ij}(t) \\ &= \sum_{k\in\mathcal{S}_{ij}} C^k_{ij}(t) + C^{UDP}_{ij}(t). \end{aligned} \tag{6}$$

2.3. Switch Architecture

We will distinguish two types of switches. First, an $N \times N$ *output-queued (OQ)* switch (Figure 2) 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 3) 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 work include:

- *iSLIP*, a round-robin-based algorithm [29, 28]. 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)* server 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 [38, 27, 11]. 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 $Q(t) = B$.

2.4. Single Output-Port Server 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 model, i.e. as a *server* model. This server model uses 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, in the server model, the iSLIP switch scheduling algorithm reduces to the round-robin (RR) server. Likewise, the MWM switch scheduling algorithm reduces to the longest queue first (LQF) server.

In Section 7 we will suggest how the results for the single-port $N \times 1$ *server* model can be extended to the general $N \times N$ *switch* model.

3. Fairness of OQ servers

In the next sections,we want to compare OQ and IQ servers from the point of view of fairness. To do so, we first define two simple fairness measures: Jain's fairness and utility-based TCP fairness. Then, when considering the single-output case, we show that OQ servers are *fair*, essentially stating that two flows sharing the same link should receive the same bandwidth.

3.1. Fairness Measures

Our objective is to analyze the fairness of OQ and IQ servers, i.e. the way in which the available output link capacity is divided between flows. We first define two fairness measures, and then apply these measures to compare the performance of the servers. First, we define *Jain's fairness index* [23]:

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

$$F \triangleq \frac{\left(\sum_{i=1}^{m} C_i\right)^2}{m \cdot \sum_{i=1}^{m} C_i^2} \tag{7}$$

Next, we define the *utility function* [24, 37] of each TCP flow k, and the resulting *TCP-fair resource allocation.*

Definition 2 (TCP-Fair Resource Allocation). *The utility function of a TCP flow k is*

$$U_k(C^k) \triangleq -\frac{2}{(RTT^k)^2} \cdot \frac{1}{C^k} \tag{8}$$

Let $\mathcal{S}_{\cdot j}$ be the set of flows k that share output link j of capacity C_{out}. Then a TCP-fair resource allocation is a resource allocation that achieves

$$\begin{aligned} \max \quad & \sum_k U_k(C^k) \\ \text{s.t.} \quad & \sum_{k \in \mathcal{S}_{\cdot j}} C^k \leq C_{out} \quad \forall j \\ & C^k \geq 0 \quad \forall k. \end{aligned} \tag{9}$$

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 number of packets in the queue.*

3.2. Fairness Analysis

We will now analyze the fairness measure of OQ servers, and later compare it with IQ servers. For simplicity, we consider the *server model*, 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 [24, 25, 31, 33, 34, 37] (where [33, 34] discuss droptail buffers more specially):

$$C^k = \frac{\sqrt{2}}{RTT^k \cdot \sqrt{d^k}} \tag{10}$$

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 two properties, which follow from [24], show that the throughput of all flows in the output-queued server is divided equally at the output link.

Property 1 (OQ Server Throughput). *In the OQ server defined above, the throughput of flow k is:*

$$C^k = \frac{C_{out}}{\sum_{i=1}^{N} m_i} \tag{11}$$

Proof By Assumption 4, all flows have the same dropping probability. Therefore, using the steady-state throughput of a TCP flow, as defined in Equation (10), all flows have the same throughput. Finally, by Assumption 3, we have $\sum C^k = C_{out}$, hence the result. □

Example 1. *As illustrated in Figure 4, consider a 2×1 OQ server 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*

$$C^k = \frac{C_{out}}{m_1 + m_2} = \frac{C_{out}}{11}, \tag{12}$$

independently of its input.

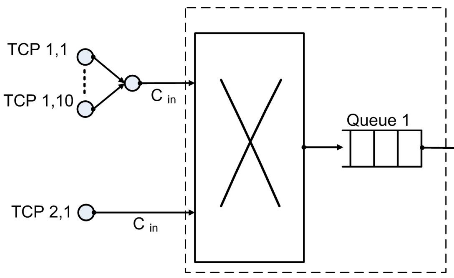


Figure 4: 2×1 OQ server example

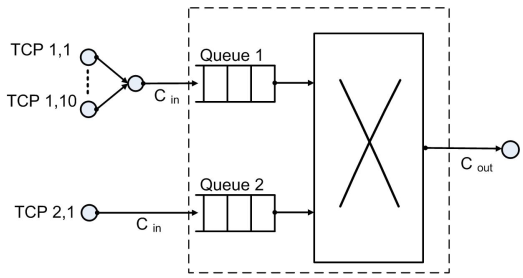


Figure 5: 2×1 IQ server example

Property 2. *The OQ server* maximizes all fairness measures*:*
(i) *Jain's fairness index achieves its maximum $F = 1$, and*
(ii) *The total utility function achieves its maximum as well, therefore the resource allocation is TCP-fair.*

Proof Since all flows share the total throughput equally, Jain's fairness is clearly equal to 1: if $m = \sum_{i=1}^{N} m_i$, then

$$F = \frac{\left(m \cdot \frac{C_{out}}{m}\right)^2}{m \cdot m \cdot \left(\frac{C_{out}}{m}\right)^2} = 1. \tag{13}$$

Further, since the total utility function is defined as a sum of equal concave functions of C^k with a single constraint on their sum, the symmetric maximal resource allocation necessarily achieves the maximum total utility function. Denote $\alpha = 2/(RTT^k)^2$. Then the total utility function is

$$\sum_{k=1}^{m} U_k(C^k) = m \cdot \left(\frac{-\alpha}{C_{out}/m}\right) = \frac{-\alpha m^2}{C_{out}} \tag{14}$$

□

4. Fairness of IQ servers with iSLIP-like Scheduling

We saw above that OQ servers are fair. We now want to analyze iSLIP-based IQ servers. Under the same server model, we prove that IQ servers 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 RR-based servers, we assume the same setting as in the analysis of OQ fairness, with a single output-port under the server model. 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 1 (Throughput). *In an $N \times 1$ server model with Round-Robin(RR) scheduling, the throughput of flow k in input i is*

$$C_i^k = \frac{C_{out}}{N \cdot m_i} \tag{15}$$

Proof The round-robin algorithm provides the same share of the output link capacity to each input. By Assumption 3, this provided service rate also corresponds to departures, and therefore all input ports have the same total departure rate and the round-robin schedule does not need to skip queues in this model. Thus, each input port behaves as an OQ server of rate C_{out}/N with m_i flows sharing the queue. The result follows from Property 1 on the OQ server throughput. □

Example 2. *Consider again the network from Example 1, this time with an $N \times 1$ server using an RR scheduler, as illustrated in Figure 5. 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 RR is unfair under both fairness measures. Its proof is presented in Appendix A.

Theorem 2. *The $N \times 1$ RR scheduler is unfair by both fairness criteria, unless all inputs have the same number of flows. Further,*
(i) *its Jain's fairness index is*

$$F = \frac{N^2}{\left(\sum_{i=1}^{N} m_i\right) \cdot \left(\sum_{i=1}^{N} \frac{1}{m_i}\right)}, \tag{16}$$

(ii) *and, using $\alpha = 2/(RTT^k)^2$, then its total utility function is*

$$\sum_{k} U_k(C^k) = \frac{-\alpha N}{C_{out}} \sum_{i=1}^{N} m_i^2. \tag{17}$$

The results above show that the throughput allocation in an IQ server using RR is generally unfair when all ports do not have the same number of flows. In particular, under the $N \times 1$ server model, they show that that the OQ server maintains fairness *among flows*, while the RR-based IQ server maintains fairness *among ports*.

5. Fairness of IQ Servers with MWM-like Scheduling

5.1. Starvation Mode vs. Oscillation Mode

We now analyze the fairness of IQ servers with MWM scheduling under the $N \times 1$ server model, where the scheduler reduces to the LQF algorithm. For simplicity, we reduce the analysis to the 2×1 case mentioned in the Introduction and shown in Figure 1. We first neglect timeouts and UDP traffic, and later take them into account.

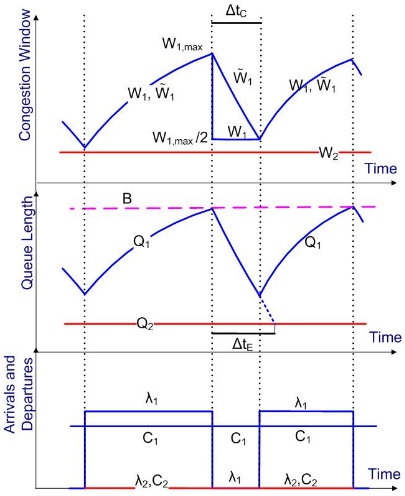


Figure 6: Starvation mode for two TCP flows in a 2×1 LQF switch

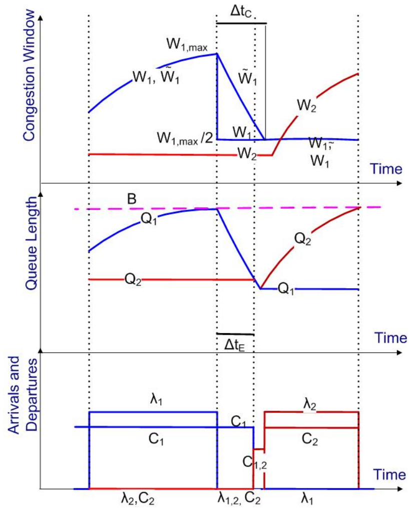


Figure 7: Oscillation mode for two TCP flows in a 2×1 LQF switch

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 LQF 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 LQF scheduler might be extremely *unfair* in such a *starvation mode*.

On the other hand, if a flow has the largest queue and keeps getting served, 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 LQF 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:

$$Q_1(t_0) > Q_2(t_0) \tag{18}$$

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 6 illustrates the typical behavior of the *starvation mode*, in which the first flow keeps prevailing and the second flow is starved. Since we always have $Q_1(t) > Q_2(t)$, the first flow keeps getting served 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* as shown in Figure 7.

In other words, during stable phases, a single prevailing queue is always being served, 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 3. *In the 2×1 LQF scheduler described above, assume that $Q_1(t_0) > Q_2(t_0)$ at time t_0. Then the server is in* starvation mode *with $Q_1(t) > Q_2(t)$ for all $t \geq t_0$ iff the buffer size B satisfies*

$$B > C_{out} \cdot \tau_1 + 2Q_2(t_0) \tag{19}$$

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

The proof is presented in Appendix B. The intuition is that if the buffers are large enough, the prevailing queue will be sufficiently long to stay longer than the starved queue after a congestion happens. On the other side, if the buffer is small, the served queue will decrease to a small value after a congestion and the other queue will receive a service, thus causing oscillation.

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 [5]. 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 LQF scheduler.

In addition, both in the *starvation* and the *oscillation modes*, we can quantify the inter-congestion time T_i of the prevailing flow i.

Theorem 4 (Inter-Congestion Time). *Both in the* starvation *and the* oscillation modes, *the inter-congestion time T_i of the* *prevailing flow i is:*

$$T_i = \frac{3 \cdot (\tau_i \cdot C_{out} + B)^2}{8 \cdot C_{out}} \tag{20}$$

Proof Appenzeller *et al.* [5] introduced the following model for the evolution of the congestion window size:

$$w^k(t) = \sqrt{C_{out} \cdot t + (w^k(0))^2}. \tag{21}$$

Assigning $w^k(0) = \frac{w_{max}}{2} = \frac{2C_{out} \cdot \tau + B}{2}$, and $w^k(T) = 2C_{out}\tau + B$, we get the result. □

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 served 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.

5.2. Starved queue length in LQF-starved mode

The maximal length of the starved queue is the minimal length of the prevailing queue. Otherwise, it contradicts the existence of starvation. This is reflected in the following result, proved in Appendix C.

Proposition 1 (Starved queue length). *The length Q_2 of the starved queue behaves according to*

$$\frac{dQ_2(t)}{dt} = \left\lfloor log_2\left(\frac{t}{RTO} + 1\right)\right\rfloor \tag{22}$$

where RTO is either the maximum Retransmission Time-Out value or the RTT of the last successful ACK as measured by the source, and

$$Q_{2,max} = Q_{1,min} \tag{23}$$

5.3. 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 served 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 5 (Equalization mode). *In the 2×1 LQF scheduler described above, the system is in* equalization mode *at time t_0 whenever the arrival rate of UDP packets λ^{UDP} is sufficiently large and satisfies*

$$\lambda_2^{UDP} > \frac{C_{out}}{Q_1(t_0) + C_{out} \cdot \tau_1} \text{ and } \lambda_1^{UDP} > \frac{C_{out}}{Q_2(t_0) + C_{out} \cdot \tau_2}.$$

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 system 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 system is either in a finite-time starvation mode *or in* oscillation mode.

The proof is presented in Appendix D. The intuition for the theorem is that above some threshold of arrival rate of UDP flow, the UDP flow has more effect on the queue length than the TCP flow. The influence of the UDP flow is more significant than the influence of the TCP flow, because the packets keep arriving, even when no packets are served in opposite to TCP.

5.4. Fairness measures of LQF server modes

We now analyze the fairness of the *starvation*, *oscillation*, and *equalization modes* in the simple 2×1 server example shown in Figure 1, where each input queue serves a single flow. In this case, MWM reduces to an LQF scheduled under a server model. The proofs of all theorems in this section are presented in Appendix E.

Starvation Mode — In this case one of the queues is always being served, 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 6. *In starvation mode, Jain's index is $F = \frac{1}{2}$.*
(i) *Jain's fairness index is $F = \frac{1}{2}$,*
(ii) *The total utility function is $\sum_i U_i(C_i) = -\infty$.*

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

Theorem 7. *Denote $\alpha_i = \tau_i \cdot C_{out} + B$. Then in oscillation mode, Jain's fairness index is $F = \frac{(\alpha_1^2+\alpha_2^2)^2}{2(\alpha_1^4+\alpha_2^4)}$.*
(i) *Jain's fairness index is $F = \frac{(\alpha_1^2+\alpha_2^2)^2}{2(\alpha_1^4+\alpha_2^4)}$.*
(ii) *The total utility function is*

$$\sum_i U_i(C_i) = \frac{-2(\alpha_1^2 + \alpha_2^2)}{C_{out}} \left(\frac{1}{\alpha_1^2 \cdot RTT_1^2} + \frac{1}{\alpha_2^2 \cdot RTT_2^2} \right) \tag{24}$$

Equalization Mode — We now characterize fairness in equalization mode.

Theorem 8. *In equalization mode,*
(i) *Jain's fairness index is* $F = \frac{(\tau_1+\tau_2)^2}{2(\tau_1^2+\tau_2^2)}$
(ii) *The total utility function is*

$$\sum_i U_i(C_i) = -\frac{2(\tau_1+\tau_2)}{C_{out}}\left(\frac{1}{(RTT_1)^2\cdot\tau_2}+\frac{1}{(RTT_2)^2\cdot\tau_1}\right)$$

UDP Mode — We now analyze the case where traffic only consists of UDP flows, without any TCP flows.

Theorem 9. *Jain's fairness index is*

$$F = \frac{1}{1+\left(\frac{\lambda_1^{UDP}-\lambda_2^{UDP}}{C_{out}}\right)^2} \tag{25}$$

Note that the total TCP utility function does not apply to this case.

6. Network Dynamics using IQ Servers

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. While the previous section characterized the long-term bandwidth averages to analyze fairness, we now focus on short-term flow dynamics.

The aim of the model is to change the known models of TCP dynamics from [24, 31, 37] given RR and LQF-based servers.

6.1. Model

We consider again the simplified *single-output* server model. We now want to describe the network dynamics in the cases of the RR and LQF-based server scheduling algorithms. To do so, we first use *many small building blocks*, which describe the behaviors of the network components (see Appendix F), and then 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 [39], and assume that the probability that flow k of input i experiences congestion given that input i experiences congestion is

$$P(t \in \mathcal{T}^k | t \in \mathcal{T}_i) = 1 - \left(1 - \frac{w^k}{C_i\cdot\tau_i + B + |\mathcal{S}_i|}\right)^{|\mathcal{S}_i|} \tag{26}$$

In the remainder, we first describe the two simplified sets of equations for iSLIP and MWM, which reduce to RR and LQF under the $N\times 1$ server model. Then, we introduce in subsequent lemmas a few interesting building blocks that were used to construct this simplified set of equations.

6.2. Network Dynamics Theorems

First, we present the dynamics of the RR network topology based on the $N\times 1$ server model. In the next theorem, the switch-scheduling equations rely on the intuition that RR equally divides output capacity among incoming ports, and divides a port capacity among flows proportionally to their number of queued packets (Equation (27)). In addition, the congestion-control equations successively model TCP flows in stable phase, TCP flows in congestion phase, and UDP traffic (Equation (28)).

Theorem 10 (RR Dynamics). *The dynamics of Internet traffic going through an* $N\times 1$ *RR scheduler 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:*

$$\begin{cases} C^k(t) = \frac{Q^k(t)}{\sum_{k'\in\mathcal{S}_i} Q^{k'}(t)+Q_i^{UDP}(t)}\cdot\frac{C_{out}}{N} & \forall i, k\in\mathcal{S}_i \\ C_i^{UDP}(t) = \frac{Q_i^{UDP}(t)}{\sum_{k'\in\mathcal{S}_i} Q^{k'}(t)+Q_i^{UDP}(t)}\cdot\frac{C_{out}}{N} & \forall i \end{cases} \tag{27}$$

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

$$\begin{cases} \frac{d}{dt}(Q^k(t)+C^k(t)\tau^k)^2 = 2C^k(t) & \textit{if } t\notin\mathcal{T}^k \\ Q^k(t^+)+C^k(t^+)\tau^k = \frac{Q^k(t^-)+C^k(t^-)\tau^k}{2} & \textit{if } t\in\mathcal{T}^k \\ \frac{dQ_i^{UDP}}{dt}(t) = \lambda_i^{UDP} - C_i^{UDP}(t) \end{cases} \tag{28}$$

The full proof of Theorem 10 is presented in Appendix F. Intuitively, Equation (27) is based on equal service rates between the input queues and is discussed further in Appendix F.3. Equation (28) is based on TCP congestion window behavior and is discussed further in Appendix F.1 and Appendix F.2. Per-flow derivation from the general per-queue equations is based on Assumption 4 and is discussed further in Appendix F.5. The derivation of the UDP flows from the general per-queue equations is also based on Assumption 4 and is discussed further in Appendix F.6.

The next theorem presents the dynamics of the LQF network topology. The switch-scheduling equations express the full service rate provided to the longest queue by LQF (Equation (30)). As in Theorem 10, the congestion-control equations model TCP and UDP flows (Equation (31)).

Theorem 11 (LQF Dynamics). *The dynamics of Internet traffic going through an $N \times 1$ LQF scheduler 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.*

$$\mathcal{A}(t) = \{i : Q_i = \max_j Q_j\}, \tag{29}$$

then

$$\begin{cases} C_i^k(t) = \frac{\sum_{k'\in\mathcal{S}_i} C^{k'}(t)+C_i^{UDP}(t)}{\sum_{k'\in\mathcal{S}_i} Q^{k'}(t)+Q_i^{UDP}(t)} \cdot Q_i^k(t) \\ C_i^{UDP}(t) = \frac{\sum_{k'\in\mathcal{S}_i} C^{k'}(t)+C_i^{UDP}(t)}{\sum_{k'\in\mathcal{S}_i} Q^{k'}(t)+Q_i^{UDP}(t)} \cdot Q_i^{UDP}(t) \\ \sum_{k\in\mathcal{S}_i} Q_i(t) = \sum_{k\in\mathcal{S}_j} Q_j(t) \quad \text{if } i,j \in \mathcal{A}(t) \\ \frac{d}{dt}\sum_{k\in\mathcal{S}_i} Q_i^k(t) = 0 \quad \text{if } i \notin \mathcal{A}(t) \\ \sum_{k=1}^m C^k(t) + \sum_{i=1}^N C_i^{UDP} = C_{out} \end{cases} \tag{30}$$

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:*

$$\begin{cases} \frac{d}{dt}(Q^k(t) + C^k(t)\tau^k)^2 = 2C^k(t) & \text{if } t \notin \mathcal{T}^k \\ Q^k(t^+) + C^k(t^+)\tau^k = \frac{Q^k(t^-)+C^k(t^-)\tau^k}{2} & \text{if } t \in \mathcal{T}^k \\ \frac{dQ_i^{UDP}}{dt}(t) = \lambda_i^{UDP} - C_i^{UDP}(t) \end{cases} \tag{31}$$

The full proof of Theorem 11 is presented in Appendix F. Equation (30) reflects the equalization of the queue sizes of the maximal queues and the lack of service for the non-maximal queues, and is discussed further in Appendix F.4. Equation (31) is similar to Equation (28), because it is based on the TCP algorithm, independently of the switch scheduling. It is discussed in Appendix F.1 and Appendix F.2. Per-flow derivation from the general per-queue equations is based on Assumption 4 and is discussed further in Appendix F.5. The derivation of the UDP flows from the general per-queue equations is also based on Assumption 4 and is discussed further in Appendix F.6.

6.3. Intuition on the $N \times N$ switch

We want to gain an intuition on the behavior of iSLIP and 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. However, we expect that their behavior will reflect the modes analyzed in $N \times 1$ switches. We provide below some heuristic rules that we would expect to see, and discuss them using simulations in Sections 7.5 and 7.6. However, because of their complexity, we do not provide any proof.

iSLIP — Consider again Assumption 3, which basically states that queues are never empty for the theoretical model. if all VOQs contain at least one flow, the switch is then expected to provide periodic TDM service between all the VOQs. We would then expect the port-fairness discussed in Section 4 to extend to a *VOQ-fairness* in the general $N \times N$ case. Therefore, the VOQs would be served equally, *independently of the number of flows in each queue*. As later shown in simulations, it appears that this intuition might even extend to some non-uniform traffic matrices in which not all VOQs contain at least one flow, yet all VOQs with at least one flow receive equal service.

MWM Equalization Mode — The MWM scheduler services the VOQs that belong to a permutation of the VOQs with the largest weight, where the weight of a VOQ is its current number of packets, and the weight of a VOQ permutation is the sum of the weights of its VOQs. Therefore, we would expect the modes to apply to *permutation sets of VOQs*, instead of applying to single queues. For instance, when an $N \times 1$ switch is in *equalization mode*, all served queues tend to be equalized. Likewise, we would expect that an $N \times N$ switch with at least one flow in each VOQ is in equalization mode when the weights of all its permutations tend to be equalized. Simulations suggest that this might in fact be extended to the case where not all VOQs necessarily contain at least one flow.

MWM Starvation Mode — The $N \times 1$ switch is in *starvation mode*, when only one of the queues is served and other queues are starved. We would expect that an $N \times N$ switch is in starvation mode when a single permutation has a weight higher than the others, and therefore is always being served. Note that some of the other permutation weights increase despite the fact that they are not served, because the permutations share VOQs with the served permutation. 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 single queue, these are now the dynamics of all packets queued in a specific permutation. (Note that we skip the oscillation mode, which is complex to analyze in a general $N \times N$ switch.)

7. Simulations

7.1. 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 RR- and LQF- based server models.

In our simulations, we modeled iSLIP and MWM switches in ns2 and used default ns2 protocol implementations of other components. For $i, j = 1, \ldots, N$ and $k = 0, \ldots, |\mathcal{S}_{i,j}| - 1$ we assumed that the round-trip propagation time of flow k at input i and output j is

$$\tau_{ijk} = \left(i + j + \frac{k}{|\mathcal{S}_{i,j}|}\right) \cdot \frac{\tau_{000}}{2}, \tag{32}$$

where $\tau_{000} = 100$ ms (unless stated otherwise) is a base propagation time and $|\mathcal{S}_{i,j}|$ is the number of flows at input i and output j. We also assumed a uniform packet size of 1 kB.

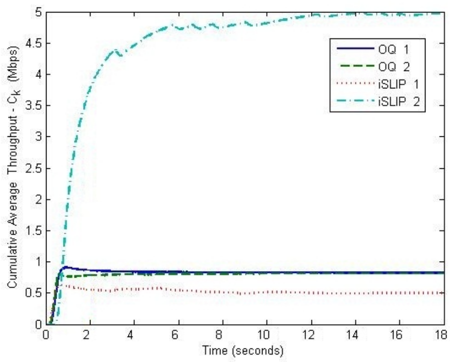


Figure 8: Simulation of OQ and RR cumulative average throughput. As expected, the graphs of OQ_1 and OQ_2 become hard to distinguish

7.2. Fairness of OQ and RR Servers

Figure 8 displays simulation results for the 2×1 switch example under the server model with 11 flows, as shown in Figure 4 and discussed in Examples 1 and 2. It plots the cumulative average throughput of one flow from each input, assuming both OQ and RR in the server model. 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 and validates Property 1 and Theorem 1. In the OQ server, the throughput of different flows equalizes over time even if they are from different inputs, thus resulting in a *fair allocation*. However, in the RR server, 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*.

By setting the simulation settings in Equation (11) we receive:

$$C^{1,2} = \frac{C_{out}}{\sum_{i=1}^{N} m_i} = \frac{10}{10+1} = \frac{10}{11} Mbps \tag{33}$$

and to Equation (15):

$$C_1^k = \frac{C_{out}}{N \cdot m_1} = \frac{10}{2 \cdot 10} = \frac{1}{2} Mbps \tag{34}$$

and

$$C_2^k = \frac{C_{out}}{N \cdot m_2} = \frac{10}{2 \cdot 1} = 5Mbps \tag{35}$$

7.3. LQF Modes

Figures 9(a), 9(c) and 9(b) show the evolution of the instantaneous queue lengths of each input in the three LQF modes, assuming the 2×1 server model. All these figures were obtained using the same switch architecture, but different network topology conditions (different buffer sizes, propagation times, and output capacity).

Figure 9(a) shows the *starvation mode*, where queue 1 is the prevailing served queue and queue 2 is the starved queue. It used a single flow per input, no UDP packets, $C_{out} = 1$ Mbps and $B = 41$ kB.

Figure 9(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 $t = 56.5$ seconds and $t = 57.5$ 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 9(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 3 and 5. For instance, in the starvation mode settings, the following condition of Theorem 3 holds:

$$\begin{aligned} C_{out} \cdot \tau_1 + 2Q_2(t_0) &\approx 1 \cdot 10^6/8 \cdot 0.1 + 2 \cdot 14 \cdot 10^3 \\ &= 40.5\text{kB} < 41\text{kB} = B \end{aligned}$$

In the oscillation mode settings, the following condition of Theorem 3 holds:

$$\begin{aligned} C_{out} \cdot \tau_1 + 2Q_2(t_0) &\approx 5 \cdot 10^6/8 \cdot 0.1 + 2 \cdot 85 \cdot 10^3 \\ &= 232.5\text{kB} > 150\text{kB} = B \end{aligned}$$

And finally, in the equalization mode settings, the following condition of Theorem 5 holds:

$$\lambda^{UDP} = 0.2 \cdot 2 \cdot 10^3 > \frac{2 \cdot 10^3}{31 + 2 \cdot 10^3 \cdot 0.1} = \frac{C_{out}}{B + C_{out} \cdot \tau_1}$$

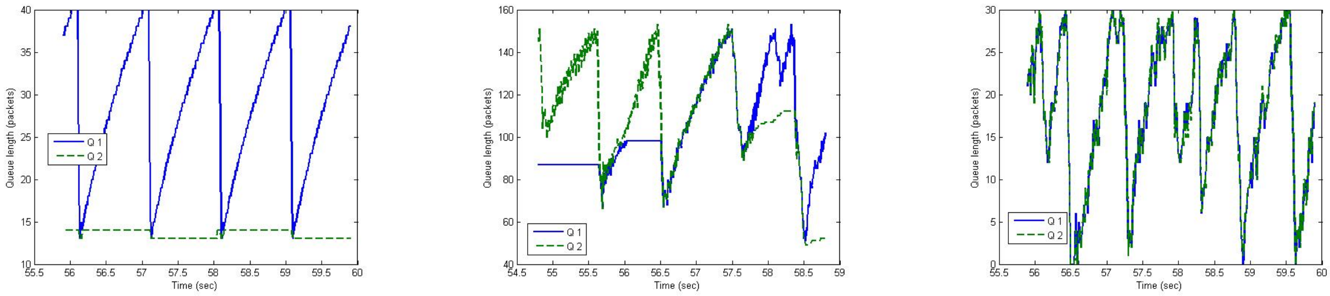
7.4. Switch Dynamics

Figures 10(a) and 10(b) show the modeled dynamics (Theorem 10) and the ns2 simulation results of the 2×1 RR server under the server model with 100 TCP flows per input, using $C_{out} = 100$ Mbps, a total UDP rate of $5\% \cdot C_{out}$ and $B = 180$ kB. On the graphs, $Q_{1,2}$ represents the sizes of each of the queues and $C_{1,2}$ the amount of served packets in last 25 ms for each of the queues. We can observe the constant and equal service rate of both queues and the similar graphs of the queue dynamics in the model and in the simulation.

Further, Figures 11(a) and 11(b) compare the LQF server dynamics (Theorem 11) 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 $5\% \cdot C_{out}$, $\tau_{000} = 50$ ms 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.

The differences between the model and the simulations result from using probabilistic (random) values in of the model, which are expressed in Equation (26).

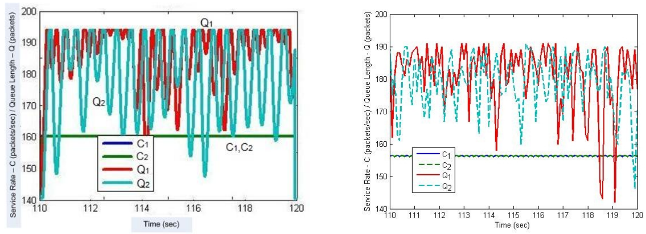


(a) LQF – Starvation mode.

(b) LQF – Oscillation mode.

(c) LQF – Equalization mode.

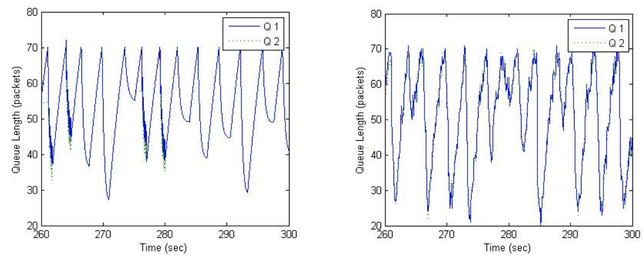
Figure 9: Fairness simulation graphs for all LQF modes in a 2×1 server model.



(a) Model.

(b) Simulation.

Figure 10: 2×1 RR server dynamics with 100 TCP flows per input. As expected, the graphs of C_1 and C_2 are hard to distinguish.



(a) Model.

(b) Simulation.

Figure 11: 2×1 LQF server dynamics with five TCP flows per input. As expected, the graphs of Q_1 and Q_2 are often hard to distinguish.

7.5. *iSLIP Scheduling in $N \times N$ Switches*

We simulated a 4×4 iSLIP switch in a network with C_{out} = 10 Mbps, B = 5000 kB, and τ_{000} = 10 ms. We used the following number of TCP flows per VOQ:

$$\begin{pmatrix} 10 & 20 & 40 & 0 \\ 30 & 50 & 30 & 0 \\ 40 & 20 & 10 & 0 \\ 0 & 0 & 0 & 50 \end{pmatrix}$$

The iSLIP scheduler served periodically 3 permutations. The service ratio of each VOQ converged to:

$$\frac{1}{3} \cdot \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 3 \end{pmatrix} \tag{36}$$

As expected, VOQ (3,3) received three times more service than other non-empty VOQs. In addition, as expected, the service is fair for VOQs, i.e independent of the number of flows in the VOQs.

In the next simulation, we used the following number of TCP flows per VOQ:

$$\begin{pmatrix} 10 & 20 & 40 & 0 \\ 0 & 50 & 30 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 50 \end{pmatrix}$$

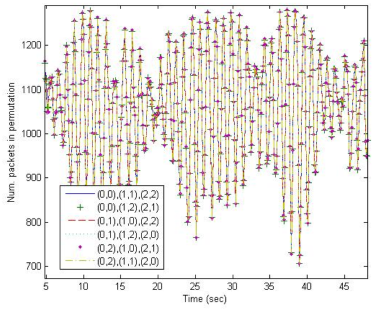
Again, the iSLIP scheduler served periodically 3 permutations. The asymptotic service ratio of each VOQ was:

$$\frac{1}{3} \cdot \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 3 \end{pmatrix} \tag{37}$$

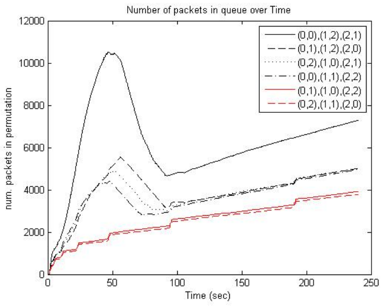
thus obtaining again a VOQ-fairness.

7.6. *MWM Modes in $N \times N$ Switches*

Figures 12(a) and 12(b) illustrate the behavior of a 3×3 MWM switch under different topology parameters. Both plot



(a) Equalization mode. The graphs appear hard to distinguish.



(b) Starvation mode.

Figure 12: Simulation graphs of 3x3 MWM switch with 100 flows per VOQ.

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 12(a) used C_{out} = 100 Mbps, B = 1.25 MB, and τ_{000} = 100 ms, while Figure 12(b) used C_{out} = 1 Mbps, B = 10 MB, and τ_{000} = 100 ms.

We can see that in Figure 12(a), the switch is in *equalization mode*, under which all permutation weights tend to stay equal. On the other hand, in Figure 12(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 the permutations include also some of the queues from the served permutation and because of UDP and timeout packets that keep arriving.

8. 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 [5, 14, 35]. In addition, it also relies on an implicit assumption that one output is more congested than others, thus yielding an $N \times N$ switch in which its influence will be predominant, and which can be modeled using a simpler $N \times 1$ switch. We leave the general model of an $N \times N$ switch to future work. However, note that in the restricted simulations above, an intuitive extension of our theory to permutations did seem to closely approximate the behavior of the simulated $N \times N$ switch.

In addition, we also assumed that the congestion only affects packets, not ACKs. This assumption is too restrictive, and ACK congestion is left for future study.

We readily acknowledge that this work does not cover all possible topologies. In fact, as further discussed in [14], while these assumptions are commonly made in the literature, they have not been substantiated enough in the past and would certainly require more study.

Similar round trip times — Assumption 2 neglects the RTT variations between flows in the same input port to keep the equations simple. In simulations, we did make sure to simulate different ranges of RTTs in order to make sure that the variance in RTTs does not significantly the results. In fact, we used a uniformly-distributed RTT distribution in each input port. In the simulations of the 3x3 switch, as shown in Equation (32), we used a ratio between the highest and lowest RTT of 3.5. Yet, we only found a limited impact of the RTT variance. Likewise, in the simulations for the Nx1 switch, we only saw a small impact of 15 % on the flow capacities between the simulations and the theoretical calculations.

Non-empty queues — Assumption 3 relies on non-empty queues in the iSLIP switch. The assumption is only needed to build a theoretical model, and is obviously wrong in the general case. We found that queues are seldom empty when buffer sizes are large enough. For instance, in our simulations, queues were typically empty less than 1 % of the time. Also, in an iSLIP switch, when a queue gets empty and does not use its assigned service, the next queue in the round-robin order is typically served.

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.

9. Conclusions

In this paper we modeled the interactions of user-based congestion control algorithms and router-based switch scheduling algorithms. 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.

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. 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 [29]. 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.

References

[1] 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.

[2] 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.

[3] M. Ajmone Marsan, P. Giaccone, E. Leonardi, and F. Neri, "On the stability of local scheduling policies in networks of packet switches with input queues," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 4, pp. 642-655, May 2003.

[4] 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.

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

[6] H. Attiya, D. Hay, and I. Keslassy, "Packet-mode emulation of output-queued switches," *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '06)*, Cambridge, MA, August 2006.

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

[8] C. S. Chang, W. J. Chen, and H. Y. Huang, "On service guarantees for input buffered crossbar switches: a capacity decomposition approach by Birkhoff and von Neumann," *IEEE IWQoS'99*, pp. 79–86, London, UK, 1999.

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

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

[11] J.G. Dai and B. Prabhakar, "The throughput of data switches with and without speedup," *Proceedings of IEEE INFOCOM*, vol. 2, pp. 556–564, Tel Aviv, Israel, March 2000.

[12] S. Deb and R. Srikant, "Rate-based versus queue-based models of congestion control,", *ACM Sigmetrics*, June 2004.

[13] 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.

[14] S. Floyd and E. Kohler, "Internet Research Needs Better Models," *First Workshop on Hot Topics in Networks*, Princeton, NJ, Oct. 2002

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

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

[17] Y. Ganjali, A. Keshavarzian, and D. Shah, "Input queued switches: cell switching vs. packet switching," *Infocom '03*, vol. 3, pp. 1651–1658, Mar. 2003.

[18] 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.

[19] 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.

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

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

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

[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.

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

[25] J. Lee, S. Bohacek, J. P. Hespanha and K. Obraczka, "A study of TCP fairness in high-speed networks," *Technical Report*, USC, July 2005.

[26] T. T. Lee, and C. H. Lam, "Path switching-a quasi-static routing scheme for large scale ATM packet switches," *IEEE Journal on Selected Areas of Communications*, vol. 15, pp. 914–924, 1997.

[27] 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.

[28] N. McKeown and T. E. Anderson, "A quantitative comparison of scheduling algorithms for input-queued switches," *Computer Networks and ISDN Systems*, 1998.

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

[30] 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.

[31] J. Padhye, V. Firoiu, D. Towsley and J. Kurose, "Modeling TCP throughput: a simple model and its empirical validation," *ACM SIGCOMM*, 1998.

[32] R. S. Prasad, C. Dovrolis, and M. Thottan, "Router buffer sizing revisited: the role of the input/service rate ratio," *ACM CoNext*, New York, 2007.

[33] G. Raina, D. Towsley and D. Wischik, "Part II: Control theory for buffer sizing," *ACM SIGCOMM Computer Communication Review*, Jul. 2005.

[34] G. Raina and D. Wischik, "Buffer sizes for large multiplexers: TCP queuing theory and stability analysis," in *Proc. Next Generation Internet Networks*, 2005.

[35] M. Shifrin and I. Keslassy, "Modeling TCP in small-buffer networks," *Networking*, Singapore, May 2008.

[36] R. Shorten, F. Wirth and D. Leith, "Modelling TCP congestion control dynamics in drop-tail environments," *Automatica*, 2007.

[37] R. Srikant, "Models and methods for analyzing Internet congestion control algorithms," In *Advances in Communication Control Networks*, Springer-Verlag, 2004.

[38] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks, *IEEE Trans. Automatic Control*, vol. 37, no. 12, pp. 1936-1948, Dec. 1992.

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

[40] T. Weller, and B. Hajek, "Scheduling nonuniform traffic in a packet switching system with small propagation delay," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 813–823, 1997.

[41] D. Wischik and N. McKeown, "Part I: Buffer sizes for core routers," *ACM SIGCOMM Computer Communication Review*, Jul. 2005.

Appendix A. Proof of Theorem 2

Proof Jain's fairness index is:

$$\begin{aligned} F &= \frac{\left(\sum_{i=1}^{N}\sum_{k=1}^{m_i}\frac{C_{out}}{N\cdot m_i}\right)^2}{\left(\sum_{i=1}^{N} m_i\right)\cdot\left(\sum_{i=1}^{N}\sum_{k=1}^{m_i}\left(\frac{C_{out}}{N\cdot m_i}\right)^2\right)} \\ &= \frac{\left(\sum_{i=1}^{N} m_i\cdot\frac{1}{m_i}\right)^2}{\left(\sum_{i=1}^{N} m_i\right)\cdot\left(\sum_{i=1}^{N} m_i\cdot\frac{1}{m_i^2}\right)} \qquad \text{(A.1)} \\ &= \frac{N^2}{\left(\sum_{i=1}^{N} m_i\right)\cdot\left(\sum_{i=1}^{N}\frac{1}{m_i}\right)} \end{aligned}$$

By the Cauchy-Schwarz inequality,

$$N^2 = \left(\sum_{i=1}^{N} \sqrt{m_i} \cdot \frac{1}{\sqrt{m_i}}\right)^2 \leq \left(\sum_{i=1}^{N} m_i\right) \cdot \left(\sum_{i=1}^{N} \frac{1}{m_i}\right), \tag{A.2}$$

i.e. $F \leq 1$, with equality iff all the m_i are equal. Likewise, the total utility function is

$$\begin{aligned} \sum_k U_k(C^k) &= \sum_{i=1}^{N} \sum_{k=1}^{m_i} \left(-\frac{\alpha}{C^k}\right) \\ &= -\alpha \sum_{i=1}^{N} \sum_{k=1}^{m_i} \frac{N \cdot m_i}{C_{out}} \\ &= \frac{-\alpha N}{C_{out}} \sum_{i=1}^{N} m_i^2 \\ &= \frac{-\alpha}{C_{out}} \left(\sum_{i=1}^{N} 1^2\right)\left(\sum_{i=1}^{N} m_i^2\right) \\ &\overset{(a)}{\leq} \frac{-\alpha}{C_{out}} \left(\sum_{i=1}^{N} 1 \cdot m_i\right)^2 \\ &= \frac{-\alpha m^2}{C_{out}}, \end{aligned} \tag{A.3}$$

where (a) follows from the Cauchy-Schwarz inequality, with equality iff all the m_i are equal.

□

Appendix B. Proof of Theorem 3

We prove Theorem 3 by first proving three lemmas on the general dynamics of flow k, and then using them to characterize the needed conditions for the starvation mode.

The first lemma characterizes the dynamics of the window size of flow k. It distinguishes the congestion times $\left(t \in \mathcal{T}^k\right)$ in which the window size is halved, the times that follow congestion in which the window size does not change as long as it's less than the number of packets in the network $\left(\tilde{w}^k(t) \geq w^k(t)\right)$, and the other times in which the growth rate of the window size is $1/RTT^k(t)$.

Lemma 1. *The congestion window size $w^k(t)$ of flow k is approximated by:*

$$\begin{cases} w^k(t^+) = \frac{w^k(t^-)}{2} & \text{if } t \in \mathcal{T}^k \\ \frac{dw^k(t)}{dt} = 0 & \text{if } \tilde{w}^k(t) > w^k(t), t \notin \mathcal{T}^k \\ \frac{dw^k(t)}{dt} = \frac{1}{RTT^k(t)} & \text{if } \tilde{w}^k(t) \leq w^k(t), t \notin \mathcal{T}^k \end{cases} \tag{B.1}$$

Proof These equations are based on the behavior of the TCP-Reno protocol, as explained above. □

The next lemma characterizes the arrival rate of flow k.

Lemma 2. *The arrival rate $\lambda^k(t)$ of flow k is approximated by*

$$\lambda^k(t) = \begin{cases} 0 & \text{if } \tilde{w}^k(t) > w^k(t) \\ \frac{w^k(t)}{RTT^k(t)} & \text{if } \tilde{w}^k(t) \leq w^k(t) \end{cases} \tag{B.2}$$

Proof When $\tilde{w}^k(t) > w^k(t)$, just after congestion, the source of flow k stops sending packets, therefore $\lambda^k(t) = 0$, until $\tilde{w}^k(t) \leq w^k(t)$ again. Then, there are $\tilde{w}^k(t) \approx w^k$ packets on the network (Equation (3)) distributed over a total time of RTT^k, so the sending rate is the ratio of these quantities. □

The last lemma compares the arrival rate $\lambda^k(t)$ with the service rate $C^k(t)$.

Lemma 3. *When $\tilde{w}^k(t) \leq w^k(t)$, the arrival rate $\lambda^k(t)$ of flow k follows*

$$\lambda^k(t) = C^k(t) + \frac{1}{RTT^k(t)} - \frac{dC^k(t)}{dt}\tau^k \tag{B.3}$$

and therefore

$$\frac{dQ^k(t)}{dt} = \frac{1}{RTT^k(t)} - \frac{dC^k(t)}{dt} \cdot \tau^k \tag{B.4}$$

Proof We know that $w^k(t) = Q^k(t) + C^k(t) \cdot \tau^k$ from Equations (2) and (3). Therefore, after differentiation and using Lemma 1, we get $\frac{dQ^k}{dt} = \frac{1}{RTT^k(t)} - \frac{dC^k}{dt}\tau^k$. Finally, using $\frac{dQ^k}{dt} = \lambda^k(t) - C^k(t)$ (Assumption 3), we get the result. □

We are finally ready to prove Theorem 3.

Proof [**of Theorem 3**] First, let's prove that $Q_1(t) > Q_2(t)$ in the stable phase following t_0 and preceding congestion. Assume by contradiction that this is only the case until time $t_0 + \Delta t$. By the proof of Lemma 3, $\frac{dQ^k}{dt} = \frac{1}{RTT^k(t)} - \frac{dC^k}{dt}\tau^k$. If until $t_0 + \Delta t$ the first flow is served at rate C_{out} while the second flow does not receive any service, then the queues keep growing at the same rate, because round-trip times are presumed equal. Therefore $Q_1(t_0 + \Delta t) > Q_2(t_0 + \Delta t)$ and there is contradiction.

Next, let's analyze the congestion phase. By Lemma 1, before congestion at time $T \in \mathcal{T}^k$, the window size w_1 is at its maximum and equal to the maximum number of packets of flow 1 in the network:

$$w_1(T^-) = w_{1,max} = C_{out} \cdot \tau_1 + B. \tag{B.5}$$

During congestion the window size is halved and $\tilde{w}_1(T^+) \approx w_1(T^-) > w^1(T^+)$. The congestion period ends when $\tilde{w}_1(t) \approx w_1(t)$ again, i.e. once $w_{1,max}/2$ packets have been transmitted at rate C_{out}. Thus the congestion period lasts

$$\Delta t_C = \frac{w_{1,max}/2}{C_{out}} = \frac{B}{2C_{out}} + \frac{\tau_1}{2} \tag{B.6}$$

Further, the time period needed to equalize queues lengths after congestion is

$$\Delta t_E = \frac{B - Q_2(t_0)}{C_{out}} \tag{B.7}$$

This is because there are $Q_1(T^-) = B$ packets of flow 1 and $Q_2(T^-) = Q_2(t_0)$ packets of flow 2 before congestion, while Q_1 decreases at rate C_{out} while Q_2 is kept constant. Therefore, to keep $\Delta t_C < \Delta t_E$, we obtain the lower-bound on B stated in the theorem. Q_2 is then kept constant during both the fluid and congestion phases, and therefore the same results can be obtained in the following stages as well. □

Appendix C. Proof of Proposition 1

Proof Based on the properties of the TCP-Reno retransmission timer. The first retransmission timer expires after the last previous measured RTT or the predefined maximum Retransmission Time-Out value (MAXRTO):

$$TO_1 \triangleq RTO = \min(RTT, MAXRTO) \tag{C.1}$$

Each next retransmission timer is twice longer than the previous one:

$$TO_i = 2 \cdot TO_{i-1} \tag{C.2}$$

So, suppose that at time $t = 0$, $Q_2(t) = 0$. Then:

$$\begin{aligned} Q_2(t) &= 1 \quad \text{if } RTO \leq t < 3 \cdot RTO \\ &= 2 \quad \text{if } 3 \cdot RTO \leq t < 7 \cdot RTO \\ &= i \quad \text{if } (2^i - 1) \cdot RTO \leq t < (2^i) \cdot RTO \end{aligned}$$

i.e.

$$Q_2(t) = i \text{ if } i \leq log_2\left(\frac{t}{RTO} + 1\right) < i + 1 \tag{C.3}$$

So,

$$\frac{dQ_2(t)}{dt} = \left\lfloor log_2\left(\frac{t}{RTO} + 1\right)\right\rfloor \tag{C.4}$$

□

Appendix D. Proof of Theorem 5

Proof Let Q_1 be the length of the served queue and Q_2 be the length of the unserved queue. The length of the served queue Q_1 is expressed by

$$\begin{aligned} \frac{dQ_1}{dt}(t) &= \lambda_1^{TCP}(t) + \lambda_1^{UDP}(t) - C_1^{TCP}(t) - C_1^{UDP}(t) \\ &= \frac{1}{RTT_1(t)} - \frac{dC_1^{TCP}}{dt}\tau_1 \end{aligned} \tag{D.1}$$

that follows from Assumption 5 and Lemma 3.

For the unserved queue, there is no service so $C_2 = 0$. Because TCP packets are not served, they are also not acknowledged, and as a result, no new TCP packets arrive, thus $\lambda_2^{TCP} = 0$ (neglecting timeout packets, as assumed previously). So the length of the unserved queue Q_2 is expressed by

$$\frac{dQ_2}{dt}(t) = \lambda_2^{UDP} \tag{D.2}$$

Equalization happens if $\frac{dQ_1}{dt}(t) < \frac{dQ_2}{dt}(t)$ for all t. Combining Equations (D.1) and (D.2) and assigning $Q_{1,max} = B$, we get the final result. □

Appendix E. Proofs of Fairness Theorems for LQF

Appendix E.1. Starvation Mode

Proof [of Theorem 6] First, Jain's fairness index is: $F = \frac{C_{out}^2}{2C_{out}^2} = \frac{1}{2}$. Further, the total utility function is $\sum_i U_i(C_i) = -\frac{2}{(RTT_1)^2}\frac{1}{C_1} - \frac{2}{(RTT_2)^2}\frac{1}{C_2}$, with $C_2 = 0$. □

Appendix E.2. Oscillation Mode

Proof [of Theorem 7] Using the inter-congestion times from Theorem 4, $C_1 = C_{out}\frac{T_1}{T_1+T_2}$ and $C_2 = C_{out}\frac{T_2}{T_1+T_2}$, and therefore $C_1 = C_{out}\frac{\alpha_1^2}{\alpha_1^2+\alpha_2^2}$ and $C_2 = C_{out}\frac{\alpha_2^2}{\alpha_1^2+\alpha_2^2}$. We conclude by replacing C_1 and C_2 in fairness Equations (7) and (8). □

Appendix E.3. Equalization Mode

We first prove the following lemma, before characterizing fairness in equalization mode.

Lemma 4. *In equalization mode, the approximate service rate C_i of queue i is*

$$C_i = C_{out}\frac{\tau_j}{\tau_i + \tau_j}, \tag{E.1}$$

where $i \neq j$.

Proof In equalization mode, $Q_i = Q_j$. Using $Q_i = W_i - C_i * \tau_i$ and $C_i + C_j = C_{out}$, we get

$$C_i = \frac{W_i - W_j + C_{out} \cdot \tau_j}{\tau_i + \tau_j} \tag{E.2}$$

Approximating the average value of W_i (and W_j) as the average between its maximum and minimum values, where the minimum equals half the maximum,

$$W_i \approx \frac{W_{i,max} + W_{i,max}/2}{2} = \frac{3(B + C_i\tau_i)}{4} \tag{E.3}$$

Substituting into the previous equation, we get the result. □

Proof [of Theorem 8] By assigning C_1 and C_2 from Lemma 4 into Equations (7) and (8). □

Appendix E.4. UDP Mode

Lemma 5. *In a* 2×1 *LQF server* with only UDP flows *the approximated service rate C_i of queue i is*

$$C_i = \frac{C_{out} + \lambda_i^{UDP} - \lambda_j^{UDP}}{2} \tag{E.4}$$

where $i \neq j$.

Proof Assume $\lambda_i^{UDP} > \lambda_j^{UDP}$. The amount $\Delta\lambda_{i,j}^{UDP} = \lambda_i^{UDP} - \lambda_j^{UDP}$ is the first to depart from queue i. The rest $(C_{out} - \Delta\lambda_{1,2}^{UDP})$ is divided equally between the queues:

$$C_i = \Delta\lambda_{1,2}^{UDP} + \frac{C_{out} - \Delta\lambda_{1,2}^{UDP}}{2} \tag{E.5}$$

C_j is found using $C_j = C_{out} - C_i$. □

Proof [of Theorem 9] By assigning C_1 and C_2 from Lemma 5 into Equation (7). □

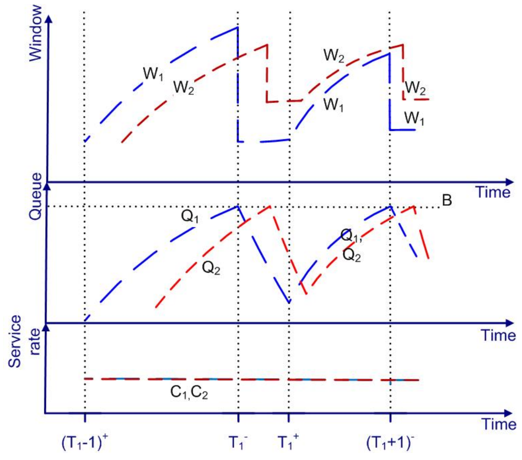


Figure F.13: 2×1 RR model.

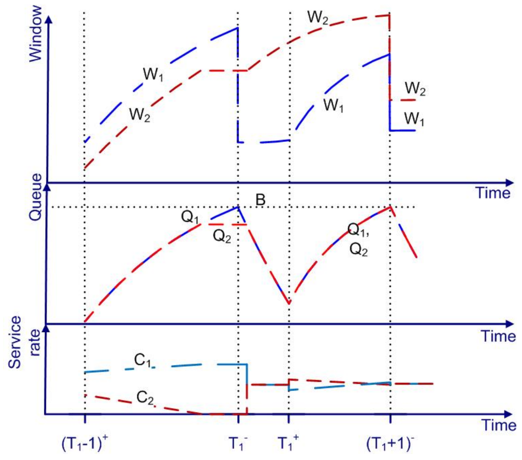


Figure F.14: 2×1 LQF server model.

Appendix F. Proofs of Network Dynamics Theorems

In the proofs below, for each congestion time $T \in \mathcal{T}_i$ we distinguish two different times for simplicity. Let $t = T^-$ be the time when the congestion of the flow happened and $t = T^+$ be the time when the flow recovered from the congestion.

Figures F.13 and F.14 illustrate the notations and behavior of the dynamics model for RR and for LQF server. The model composed of two phases: the *stable phase* and the *congestion phase*.

The *stable phase* models the stable situation, when queues are not full. The output link is fully utilized, the congestion windows are continuously increasing (because there are no drops), so the queues are continuously increasing too (Equations (2) and (3)). During the *stable phase* of queue i, the arrival rate exceeds the service rate, i.e. $\lambda_i \geq C_i$.

The *congestion phase* starts immediately after the congestion indication. Following congestion, there is window halving: the source waits for packets to be acknowledged, in order to equalize the number of packets on the fly to the window size. Meanwhile, the arrival rate decreases, and therefore the queue length decreases as well, because the arrival rate is lower then the service rate. Therefore, during the *congestion phase* of queue i, $\lambda_i < C_i$.

Appendix F.1. TCP stable phase

We provide below a proof, in which we demonstrate the congestion control equations in the stable phase.

Lemma 6. *In the stable phase, each TCP flow k satisfies*

$$\frac{d}{dt}((Q^k(t) + C^k(t)\tau^k)^2) = 2C^k(t) \tag{F.1}$$

Proof First, all flows k in the same input i have the same round-trip time $RTT_i(t) \triangleq RTT^k(t) = \tau^k + \frac{Q^k(t)}{C^k(t)}$, because they have the same $\tau^k(t)$ (Assumption 2) and because their share of the input bandwidth ($C^k(t)$) is proportional to their share of the queue size ($Q^k(t)$) (Assumption 4). Next, in the stable phase, from Equations (2) and (3), the congestion window size of each flow k is assumed to satisfy $w^k(t) = C^k(t) \cdot RTT^k(t)$. Further, for each flow k, $\frac{dw^k(t)}{dt} = \frac{1}{RTT^k(t)} = \frac{1}{RTT_i(t)}$, thus $\frac{d}{dt}(Q^k(t)+C^k(t)\tau_i)^2 = \frac{d}{dt}w^k(t)^2 = 2w^k(t) \cdot \frac{d}{dt}w^k(t) = 2\left(C^k(t) \cdot RTT^k(t)\right) \cdot \frac{1}{RTT^k(t)}$, hence the result. $\square$

Appendix F.2. TCP congestion phase

Next is the set of m equations of the TCP congestion phase.

Lemma 7 (TCP congestion phase equations). *In the congestion phase, each TCP flow k satisfies*

$$Q^k(T^+) + C^k(T^+) \cdot \tau^k = \frac{Q^k(T^-) + C^k(T^-) \cdot \tau^k}{2} \tag{F.2}$$

Proof Resulting immediately from the properties of TCP-Reno and based on the dynamics of w^k that are shown in Lemma 1:

$$\begin{aligned} w^k(T^-) &= Q^k(T^-) + C^k(T^-) \cdot \tau^k \\ w^k(T^+) &= \tfrac{w^k(T^-)}{2} \end{aligned} \tag{F.3}$$

$\square$

Appendix F.3. RR switch scheduling

The service rate of queue i in an RR switch is defined by the number of input ports (N), as shown in the following lemma.

Lemma 8 (RR server equations). *The service rate $C_i(t)$ of input queue i in an $N\times1$ RR switch is defined by the next equation:*

$$C_i(t) = \frac{C_{out}}{N} \tag{F.4}$$

Proof The service rate of the switch is divided equally between the inputs, as showed in Theorem 1. $\square$

Note that the $N \times 1$ RR server with service rate C_{out} can be modeled as N FIFO queues with service rate C_{out}/N, as each queue receives an equal service rate independently of each other.

Appendix F.4. LQF switch scheduling

We denote $\mathcal{A}(t)$ as the set of active queues at time t, i.e. queues with $C_i(t) > 0$. Then using LQF, all queues $i, j \in \mathcal{A}(t)$ tend to have equal length: $Q_i(t) = Q_j(t)$ at time t.

Lemma 9 (Queue equalization). *If* $0 < C_i(t), C_j(t) < C_{out}$, *the lengths of the queues* i, j *in a* $N \times 1$ *LQF switch behave according to*

$$Q_i(t) = Q_j(t) \tag{F.5}$$

Proof $N \times 1$ LQF scheduling gives service to the longest queue. If at some time several queues were served, they all have the longest queue length, and therefore can be modeled as equal. □

Next is the set of the N equations of the LQF switch scheduling.

Lemma 10 (LQF switch scheduling equations).

$$\begin{cases} Q_i(t) = Q_j(t) & \text{if } i, j \in \mathcal{A}(t) \\ dQ_i^{TCP}(t) = 0 & \text{if } i \notin \mathcal{A}(t) \\ \sum_{i=1}^{N} C_i = C_{out} \end{cases} \tag{F.6}$$

Proof Resulting from Lemma 9 and the TCP-Reno properties. □

Appendix F.5. Per-flow equations

The service rate of each TCP flow satisfies the following lemma.

Lemma 11 (Per-flow equations). *Each TCP flow* k *in input* i *behaves according to the next set of equations:*

$$\begin{cases} C_i^k(t) = C_i(t) \cdot \frac{Q_i^k(t)}{Q_i(t)} \\ \frac{dQ_i^k}{dt} = \lambda_i^k(t) - C_i^k(t) = \frac{dw_i^k}{dt} - \frac{dC\tau_i}{dt} - C_i^k(t) \end{cases} \tag{F.7}$$

The dynamics of w_i^k *are shown in Lemma 1.*

Proof The first equation results from Assumption 4, stating that the service rates of two flows sharing the same queue are proportional to their queue sizes. The second equation results from the queue dynamics [5]. □

Appendix F.6. UDP equations

The next set of equations describes the influence of UDP traffic.

Lemma 12 (UDP equations).

$$\begin{cases} Q_i^{UDP} = Q_i - Q_i^{TCP} \\ C_i^{UDP} = C_i - C_i^{TCP} \\ C_i^{UDP} = \frac{Q_i^{UDP}}{Q_i} C_i \\ \frac{dQ_i^{UDP}}{dt} = \lambda_i^{UDP} - C_i^{UDP} \end{cases} \tag{F.8}$$

Proof The first two equations result from the definitions of Q_i^{UDP} and C_i^{UDP}. As in Lemma 11, the third equation comes from Assumption 4, stating that the service rates of two flows sharing the same queue are proportional to their queue sizes. The last equation comes from Assumption 3. □