Rules for Faster Convergence to Fairness

A.Tsioliaridou[a], Chi Zhang[b], V.Tsaoussidis[a]

[a] *Department of Electrical and Computer Engineering*
Democritus University of Thrace, Greece
(atsiolia,vtsaousi)@ee.duth.gr
[b] *Juniper Networks, USA*
chizhang@juniper.net

Abstract

We extend the theoretical model of Chiu and Jain described in [2] by taking into account the role of bottleneck buffer and present a new analysis that allows system to reach equilibrium fast, based on throughput measurements. In our model, no flow is aware of the number of competitors in the channel, no flow is aware of the bandwidth and buffer capacity; however, at the end of each epoch, which is signaled by a congestion event, each flow can i) estimate whether it has operated above, beyond or around its fair-share and ii) adjust its window to its fair-share. Within this context, we present two rules on how to adjust the transmission rate after a congestion event, namely Fair-Share and Fairness rule. They both promote fairness without damaging system efficiency and responsiveness. Simulation results confirm the validity of our analysis and the proposed congestion control schemes.

Key words:
fairness, fair-share, throughput analysis, congestion control

1. Introduction

In a shared network, such as the Internet, flows should react to congestion by adapting their transmission rates to avoid congestion collapse in a manner that the total bandwidth of the network will be expended fairly and effectively. This responsive behavior entails several challenges. Firstly, there is no centralized system to control the network traffic. Moreover, network flows do not have any prior or present knowledge of network characteristics, such as the number of competing flows, links bandwidth, routers queue size etc. Despite these limitations, the varying number of participating flows, must be adaptive to network changes.

Transmission control of standard `TCP` [3], [4] is based on the Additive Increase/ Multiplicative Decrease (`AIMD`[2]) window adjustment strategy. `AIMD` is a somewhat "blind" mechanism, in the sense that the congestion window increases steadily until the actual occurrence of congestion, which, in turn, necessitates error recovery. Although the linear window adjustment of `AIMD` eventually reaches an optimal point, it takes several epochs to reach equilibrium. Should realistic, rapid chances occur (e.g. a change in the number of flows), the system may not even reach equilibrium, due to long duration of the convergence period. Even when convergence has been achieved, the equilibrium satisfies only instantly the notion of fairness. That is, although we managed to bring an unfair system to a state of fairness, we did not manage to allocate resources fairly throughout the *lifetime* of our system.

This observation prompts us to establish further criteria for judging fairness. For example, the duration required for a system to reach equilibrium reflects the duration the system was unfair and, in this context, reflects the level of system unfairness. Additionally, our observation calls for measures to adjust fairness from an instance of time to a timescale that corresponds to the lifetime of the system in turn, it calls for establishment rules and mechanisms to allow flows to either pay back or earn further credit along the lines of their previous behavior.

In this context, first we extend the model described in[2], by taking into account the role of the routers' buffer and present a mechanism that allows each flow independently to estimate its deviation of its fair-share. More particularly, each flow becomes aware of whether it has operated beyond, below or around its fair-share, which allows them to determine the next congestion control strategy for fast convergence to fairness. We then introduce the necessary window adjustment rules. According to the Fair-Share rule each flow can estimate and instantly operate at its fair-share without damaging the network efficiency. Simulation results confirm that, in the presence of the Fair-Share rule, system converges in one congestion cycle (one epoch) to equilibrium. However, system resources have not necessarily been allocated fairly among flows, since during the period of measurements some flows may have consumed more resources than others. Therefore, we introduce the Fairness rule, which acts as an add-on and guarantees that each flow operates in association to the amount of resources that has consumed.

In the next Section, we present our system model for the theoretical throughput analysis. In Section 3, we present our analysis on throughput and, in Section 4, we discuss the deficiencies of `AIMD` algorithm on system convergence to equilibrium. The proposed window adjustment rules are detailed in Section 5 and evaluated through simulations in section 6. Finally, in Section 7, we conclude our work and suggest directions for future work.

2. System Model

Our model is initially characterized by a synchronous generation of responses, in congruity with [2]. Similarly, we assume a centralized feedback model, where all flows become aware of congestion events synchronously. In the current study, however, we extend the model described in [2] by taking into account the role of the router's buffer at the bottleneck point. Note that although we allow for the possibility of queueing delays, we consider that flows experience the congestion events almost simultaneously. Thus the duration of an `epoch` is almost equal for all flows. The synchronous flows' notifications is possible in real networks, e.g. by through multiple packet dropping at the end of an epoch, when the capacity of the queueing buffer has been exhausted.

The proposed model takes into account realistic network characteristics, namely the router's uplink capacity C and buffer size `BS`; Each flow is not aware of the throughput rates (window sizes) of other flows; Each flow is not aware of the number of competitors in the channel; No flow is aware of the bandwidth `B` and buffer capacity `BS`.

Finally, the network topology used in the context of this paper is the typical dumbbell and is illustrated in Fig. 1. Table 1 summarizes the terms used throughout this paper.

3. Throughput Analysis

In this section, we study the dynamics of throughput from the flow perspective. We provide an analytical and intuitive explanation for these dynamics by concentrating on the window-based transmission control of `TCP` and by incorporating the role of the bottleneck queue. Based on throughput measurements, we show that at the end of an `epoch`, each flow can estimate whether it has operated beyond, below or close to its fair-share.

Flows are assumed to initially follow the Additive Increase / Multiplicative Decrease rule. Figure 2 shows the throughput of a system during an

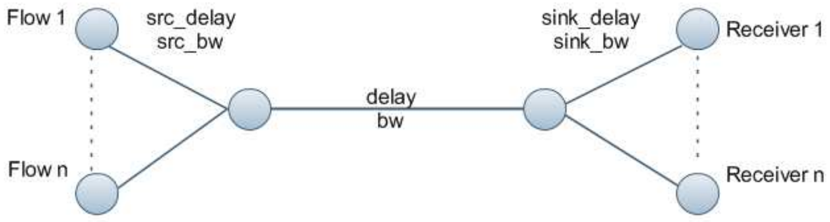


Figure 1: Simple network topology

Symbol	Meaning
cwnd	Current congestion window
t	Time of sampling
epoch	The interval between two cwnd multiplicative decreases
i	Round number
B	The Bandwidth of the Uplink
C	The Capacity of the Uplink
BS	Router's Buffer size
ACK	Acknowledgement Packet
RTT	Round Trip Time

Table 1: Algorithm symbols

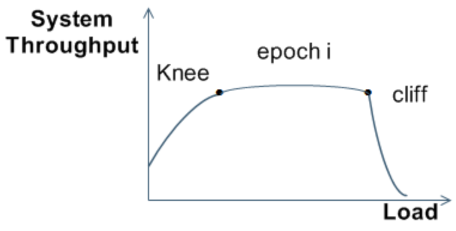


Figure 2: System Throughput

`epoch`. As the number of packets inserted into the link increases, throughput increases. After the capacity of the link is fully utilized, a queue starts to build up at the bottleneck point (router) while throughput stops increasing. Further increase of incoming rate of packets potentially results in packet drops and the network state changes from available to "congested". The point at which queue starts to build up is called *knee*. Since flows adjust their window simultaneously, the *knee* point may be detected by all flows from `RTT` measurements. It is precisely the point where the `RTT` value is greater than the RTT_{min}, which corresponds to the Round Trip propagation delay.

Throughput Analysis. Consider a system described in Section 2, where `n` users share a bottleneck link with capacity `C` and round trip delay RTT_o. The flows' data rates are gradually increasing (additive increase) and the network eventually becomes congested. The network signals the senders about the change of the state (from available to "congested") and the senders reduce their sending rate. Throughput for the i^{th} flow at time `t` is defined as:

$$Throughput = \frac{cwnd_i(t)}{RTT(t)} = \frac{cwnd_i(t)}{RTT_o + qdelay_i(t)} \tag{1}$$

where $cwnd_i(t)$ is the congestion window of the i^{th} flow at time `t` and $qdelay_i(t)$ is the corresponding queueing delay at the bottleneck router at time `t`. Note that Throughput is not only a function of the congestion window, but also a function of the dynamic queueing delay, which is not incorporated into the analysis of [2], [5].

Assume that all flows are in the additive increase stage. If the system operates below the *knee* point, then there is no *steady* queue buildup [1] at the router, and according to Equation (15), throughput of each flow grows in proportion to its `cwnd`, since the bottleneck capacity is not fully utilized. If the system load increases further beyond `bw`, flows display different dynamics: the bottleneck queue starts to build up and flows experience queueing delay, until the queue length reaches the maximum buffer size (system reaches the *cliff* point of Figure 15).

Consider two measurement sample points: point `A` at time t_1 and point `B` at time t_2 between the *knee* and the *cliff*. Throughput of the i^{th} flow at time t_1 and t_2 are $throughput_i(t_1)$ and $throughput_i(t_2)$, respectively and $\Delta Throughput_i(t_2)$ is:

$$\Delta throughput_i(t_2) = throughput_i(t_2) - throughput_i(t_1) =$$

[1]There could be *temporary* queue buildup due to traffic burstiness. However, this is neglected to simplify our analysis

$$= \frac{cwnd_i(t_1) + \Delta cwnd_i(t_2)}{RTT(t_2)} - \frac{cwnd_i(t_1)}{RTT(t_1)} =$$

$$\frac{\Delta cwnd_i(t_2) RTT(t_1) - cwnd_i(t_1) \Delta q_{delay}}{[RTT_o + q_{delay}(t_2)][RTT_o + q_{delay}(t_1)]} \tag{2}$$

$\Delta qdelay$ is defined as:

$$\Delta q_{delay} = q_{delay(t_2)} - q_{delay(t_1)} =$$

$$= \frac{\sum_{k=1}^{n} cwnd_k(t_2) - \sum_{k=1}^{n} cwnd_k(t_1)}{bw} =$$

$$= \frac{\Delta \left(\sum_{k=1}^{n} cwnd_k(t_2) \right)}{bw} \tag{3}$$

Consequently from (2) and (3) $\Rightarrow$:

$$\Delta throughput_i(t_2) = \frac{\Delta cwnd_i(t_2) RTT(t_1) - cwnd_i(t_1) \frac{\Delta \left(\sum_{k=1}^{n} cwnd_k(t_2) \right)}{bw}}{[RTT_o + q_{delay}(t_2)][RTT_o + q_{delay}(t_1)]} =$$

$$= \frac{\Delta cwnd_i(t_2) RTT(t_1) bw - cwnd_i(t_1) \Delta \left(\sum_{k=1}^{n} cwnd_k(t_2) \right)}{[RTT_o + q_{delay}(t_2)][RTT_o + q_{delay}(t_1)] bw} \tag{4}$$

At this point we make the following assumption:

If network resources were allocated equally among competing flows, they would all experience the same increase of their congestion window, and equal to $\Delta cwnd_i(t_2)$ during the time period from time t_1 to t_2, defined as:

$$\Delta \left(\sum_{k=1}^{n} cwnd_k(t_2) \right) = n \Delta cwnd_i(t_2) \tag{5}$$

Consequently, equation (4) becomes:

$$\Delta throughput_i(t_2) = \frac{\Delta cwnd_i(t_2) RTT(t_1) bw - cwnd_i(t_1) \cdot n \cdot \Delta cwnd_i(t_2)}{[RTT_o + q_{delay}(t_2)][RTT_o + q_{delay}(t_1)]bw} =$$

$$= \frac{\Delta cwnd_i(t_2)}{[RTT_o + q_{delay}(t_2)]}[1 - \frac{n \cdot cwnd_i(t_1)}{RTT(t_1) \cdot bw}] \Leftrightarrow$$

$$\Delta throughput_i(t_2) = \frac{\Delta cwnd_i(t_2)}{[RTT_o + q_{delay}(t_2)]}[1 - \frac{n \cdot cwnd_i(t_1)}{\sum\limits_{k=1}^{n} cwnd_k(t_1)}] \quad (6)$$

Note that the term $\frac{\Delta cwnd_i(t_2)}{[RTT_o + q_{delay}(t_2)]}$ in equation (6) is always positive while the term $[1 - \frac{n \cdot cwnd_i(t_1)}{\sum_{k=1}^{n} cwnd_k(t_1)}]$ term might be either positive or negative. It is obvious that equation (6) allows i^{th} flow to determine whether it operates below, beyond or adjacent its fair-share. Specifically:

- If $\Delta Throughput(t_2) > 0$, the rate of i^{th} flow at time t_1 is below its fair-share, since $\sum\limits_{k=1}^{n} cwnd_k(t_1) > n \cdot cwnd_i(t_1)$
- If $\Delta Throughput(t_2) < 0$, i^{th} flow at time t_1 has reached a rate above its fair-share, since $\sum\limits_{k=1}^{n} cwnd_k(t_1) < n \cdot cwnd_i(t_1)$
- If $\Delta Throughput(t_2) = 0$, i^{th} flow has reached its fair-share, since $\sum\limits_{k=1}^{n} cwnd_k(t_1) = n \cdot cwnd_i(t_1)$

Therefore, equation (6) establishes a criterion, based on which each flow independently can review its transmission rate and deduce whether it operates greedily, fairly or suffering mistreatment.

4. Observations on the dynamics of AIMD

In this section, we investigate the operational properties of `AIMD` and extend the initial study in [2] by taking into consideration the role of the bottleneck buffer.

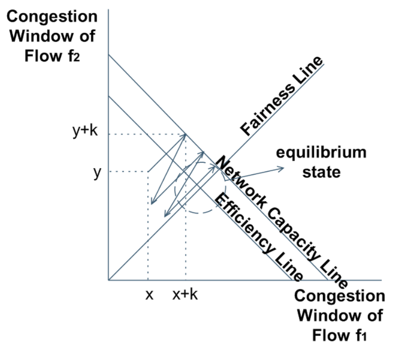


Figure 3: Vector representation of two-flow convergence to fairness

Consider the system presented in Section 2. The convergence behavior of two flows is depicted by vectors in a 2-dimensional space oscillating around the efficiency line as in Figure 3. Two flows, f_1 and f_2, have x and y initial windows, where $x < y$, $x + y < C + BS$. The horizontal axis represents the congestion window of flow f_1, while the vertical axis the congestion window of flow f_2. The values of congestion windows for which $x + y = C$, are represented by a straight line marked as "*Efficiency Line*". The values of congestion for which $x+y = C+BS$, form a straight line marked as "*Network Capacity Line*". Finally, the "Fairness Line" consists of all points for which $x = y$.

Both users follow the Additive Increase / Multiplicative Decrease control policy to reach the equilibrium area. When flows are in the additive increase phase they move parallel to the 45^o axis (Additive Increase parameter $\mathtt{a}_I$=1) and flows converge the line of *efficiency.* Once flows exceed the *Network* line, they decrease their current window using the Multiplicative Decrease parameter $\mathtt{b}_D$=1/2 and move closer to the Fairness line.

Based on the above description, we highlight three observations and arrive at conclusions that constitute the foundation of the present work:

1. When flows f_1 and f_2 are in additive increase phase, equal amount of system resources is being allocated to the flows.

2. The Multiplicative Decrease rule aims to minimize the initial windows of the two flows (x and y); at this point both flows have the same congestion window at the beginning of the next phase of Additive Increase.
3. The initial windows' values are responsible for the system's slow convergence to fairness.
4. The "distance" between the Network Capacity Capacity Line and the Efficiency Line depends only on the multiplicative decrease factor.

Practically, both fairness and efficiency can be achieved with the "appropriate" decrease of the current window. During the additive increase phase the flows increase their resource consumption uniformly, and therefore the additive increase factor should not be affected. The efficiency is associated with the utilized bandwidth and can be ensured as long as the aggregate window of flows operates between the *Efficiency* and *Network Capacity* lines. Thus window reduction should keep the load above *efficiency* line and close to *fairness* line to ensure fair allocation of network resources. This objective can be achieved only when flows operate around their fair-share. Therefore, the real issue for protocol design remains the specification of the adjustment rules for the congestion window.

5. The Congestion Window Adjustment Rules

In this section, we present two window adjustment rules to regulate the amount of data that a flow inserts into the network. Both rules rely on throughput measurements and act at the end of each `epoch`, which is determined by packet loss or the arrival of a congestion marked packet. The estimated rate adjustment applies to the next epoch (the one that is about to begin).

The first decrease rate scheme, namely the "Fair-Share rule", gives the appropriate window adjustment that ensures operation at its fair-share; while the second rule, the "Fairness rule", guarantees the equal sharing of resources among flows during the i^{th} and $(i+1)^{th}$ epoches.

5.1. The Fair-Share Rule

The Fair-Share Rule. In order to operate at its fair-share, the flow should adjust its congestion window, according to equation:

$$cwnd_{fair-share} = \frac{cwnd_B + x}{RTT_B} \cdot RTT_{\min} \tag{7}$$

where

$$x = \frac{RTT_A cwnd_B - RTT_B cwnd_A}{RTT_B - RTT_A}$$

Notations A and B represent two measurement samples during the i^{th} epoch at $time_A$ and $time_B$, respectively, where $time_A < time_B$ and $RTT_A, RTT_B > RTT_{min}$.

Justification. Flows start sending packets according to the Additive Increase Rule($a_I = 1$). When the capacity of the link is fully utilized and buffer capacity at the bottleneck link is exhausted, the system notifies all flows to decrease their windows. Based on Eq. 6, each flow can estimate whether or not it has reached its fair-share during the i^{th} epoch and reach one of the following conclusions:

- The flow has operated above its fair-share.
- The flow has operated beyond its fair-share.
- The flow has operated around its fair-share.

Next, we study each one of the above cases and calculate the appropriate value of the current window, which guarantees operation at the flow's fair-share.

1^{st} **case.** The flow has operated above its fair-share.

The throughput measurements and `cwnd` values of a flow f_1, during the i^{th}-epoch, are shown in Figure 4. Since flow f_1 has exceeded its fair-share i) throughput between the *knee* and the *cliff* points decreases and ii) the optimal line of `cwnd`, which guarantees operation at its fair-share, is lower than the current line of operation.

Consider two points, point `A` at time t_A and point `B` at time t_B between *knee* and *cliff*, at which flow f_1 received feedback (ACKs) from the receiver. Flow f_1 records the `cwnd` values, $cwnd_A$ and $cwnd_B$, measures the round trip time of packets, RTT_A and RTT_B, and calculates the corresponding values of throughput, $throughput_A$ and $throughput_B$.

Next, consider that the *optimal* values of `cwnd`, that guarantee operation at its fair-share at times t_A and t_B, are $cwnd_C$ and $cwnd_D$ respectively. Consequently, the line between the point C and D is the *fair-share* line. According to Eq.(6):

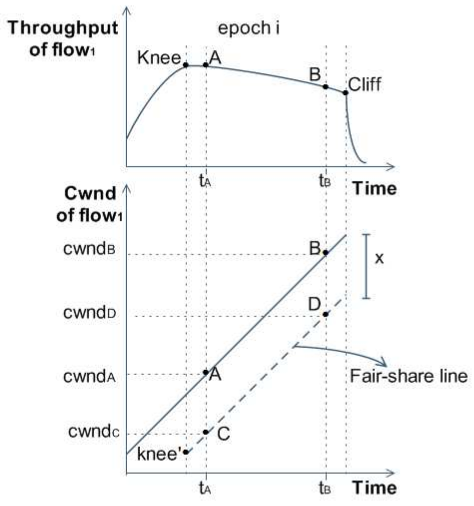


Figure 4: Flow operation above its fair-share.

$$Throughput_C = Throughput_D$$

and therefore,

$$\frac{cwnd_C}{RTT_C} = \frac{cwnd_D}{RTT_D} \tag{8}$$

Since the additive increase factor is the same in both cases ($\mathtt{a}_I = 1$), vector $\vec{CD}$ is parallel to vector $\vec{AB}$ ($\vec{CD} \,//\, \vec{AB}$), and thus it holds that $cwnd_C = cwnd_A + x$ and $cwnd_D = cwnd_B + x$. Consequently:

$$Eq.(8) \Leftrightarrow \frac{cwnd_A + x}{RTT_A} = \frac{cwnd_B + x}{RTT_B} \tag{9}$$

Note that due to different rate of incoming packets at the router during the $(i+1)^{th}$ `epoch`, the flow f_1 might not have measurement samples for which $RTT_C = RTT_A$ and $RTT_D = RTT_B$. However the *fair-share* line of the congestion window does not depend on these values, since the additive increase factor is static. Hence, from eq.(9) we derive

$$x = \frac{RTT_A cwnd_B - RTT_B cwnd_A}{RTT_B - RTT_A} \tag{10}$$

The decrease of `cwnd` at the end of the i^{th}`epoch`, should also guarantee that system efficiency is beyond the *efficiency* line. So the *optimal* point is the *knee'* point for which $RTT_{knee} = RTT_{min}$. We seek to satisfy:

$$Throughput_{knee'} = Throughput_D$$

$$\Leftrightarrow \frac{cwnd_{knee'}}{RTT_{\min}} = \frac{cwnd_D}{RTT_B} \qquad (11)$$

From equations (10) and (11) we conclude:

$$cwnd_{fair-share} = cwnd_{knee'} = \frac{cwnd_B + x}{RTT_B} \cdot RTT_{\min}$$

2nd **case.** The flow has operated below its fair-share.

The throughput and `cwnd` samples of flow f_1 are depicted in Figure 5. Since flow f_1 has consumed less resources than its fair-share, the throughput line is increasing and the optimal `cwnd` line is greater than the measured one. Similarly to before we conclude that:

$$cwnd_{fair-share} = cwnd_{knee'} = \frac{cwnd_B + x}{RTT_B} \cdot RTT_{\min}$$

where $x = \frac{RTT_A cwnd_B - RTT_B cwnd_A}{RTT_B - RTT_A}$

3rd **case.** The flow has reached its fair-share. In this case equation (7) is verified as follows: According to Equation (6):

$$Throughput_A = Throughput_B$$

$$\Leftrightarrow \frac{cwnd_A}{RTT_A} = \frac{cwnd_B}{RTT_B} \qquad (12)$$

Consequently:

$$Eq.(10) \Leftrightarrow x = 0$$

and

$$Eq.(7) \Leftrightarrow cwnd_{fair-share} = \frac{cwnd_B}{RTT_B} \cdot RTT_{\min} = cwnd_{knee}$$

which is true, since flow operates at its fair-share line, during the i^{th} `epoch`.

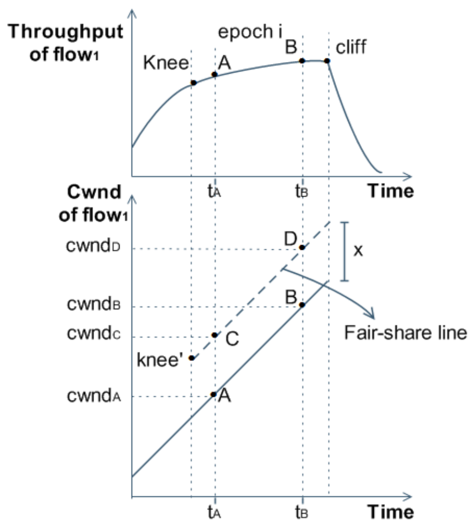


Figure 5: Flow operation below its fair-share.

5.2. The Fairness Rule

According to the Fair-Share rule, a flow can estimate its fair-share and adjust to it at the end of each `epoch`. However greedy flows that have consumed more bandwidth than others should pay back their credit, while flows that operated below their fair-share, should get more resources before final convergence to equilibrium. Therefore, we introduce the Fairness rule, to complement the Fair-Share rule. The Fairness rule compensates for any injustice for one epoch, before handing over the control to the Fair-share rule. More specifically, the Fairness rule is adopted when the fair-share of the flows has changed due to flow contention changes, or on the initialization of a flow.

The Fairness Rule. In order to achieve system equilibrium, the congestion window should be adjust according to:

$$cwnd_{fairness} = cwnd_{fair-share} \pm x \tag{13}$$

where

$$x = \frac{RTT_A cwnd_B - RTT_B cwnd_A}{RTT_B - RTT_A}$$

x is added in Equation(13) when the flow has operated below its fair-share, and subtracted in the opposite case.

A and B depict two measurement samples during the i^{th} epoch at $time_A$ and $time_B$, respectively, where $time_A < time_B$ and $RTT_A, RTT_B > RTT_{min}$.

Justification.

In the same context, during the i^{th} `epoch`, flow f_i might have operated below, above or around its fair-share.

1^{st} **case.** The flow exceeded its fair-share.

The flow has consumed more resources during the i^{th} `epoch`, equally to x packets [2] [see (10)]. Upon the reception of the congestion signal, the flow should adjust its window to x packets less than its fair-share (see Figure 6), in order for it to pay back its credit. That is:

$$cwnd_{fairness} = cwnd'_{fair-share} - x$$

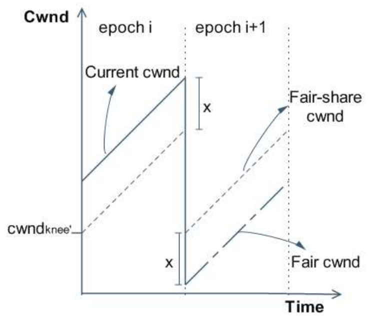


Figure 6: fair cwnd decrease

2^{nd} **case.** The flow has operated below its fair-share.

The flow has consumed less resources during the i^{th}`epoch`, equally to x packets, see Equation (10). Upon the reception of the congestion signal, the flow should adjust its window to x packets more than its fair-share(see Figure 7). That is:

$$cwnd_{fairness} = cwnd'_{fair-share} + x$$

3^{rd} **case.** The flow has operated adjacent to its fair-share.

During the i^{th}`epoch`, the flow has consumed resources that correspond to its fair-share. Upon the reception of the congestion signal, it should decrease its window according to the Fair-Share Rule.

[2]In reality this quantity is measured in bytes in `TCP`

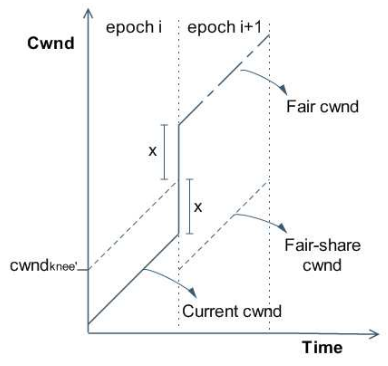


Figure 7: fair cwnd increase

$$cwnd_{fairness} = cwnd_{fair-share}$$

6. From Theory to Practice

In this section, we evaluate experimentally the throughput analysis presented in Section 3 and the validity of the window adjustment rules proposed in Section 4 on ns-2 network simulator [1]. For this reason, we incorporate the algorithms into TCP [6] and study the system's performance. The TCP version of choice in our experiments is TCP-Reno. TCP-Reno considers the network as a black box that has the capability to produce congestion signals; consequently TCP-Reno can adjust its sending rate accordingly. This functionality allows us to evaluate the proposed algorithms with precision, since the results are affected solely by the algorithms themselves and not by potentially dubious measurements (e.g. RTT).

Inline with the theoretical analysis, our TCP implementation deploys a window increase by one, as long as resource supply has not been exceeded. Each time a packet is sent, the sender records the system clock and the value of the congestion windows. Each time a corresponding ACK is returned, the sender re-reads the clock and the value of the congestion window, and computes the RTT and Throughput (Equation (2)). The RTT_{min} value is then updated and the flow adjusts its congestion window to the knee point, as described in Section 3.

At the end of each epoch, which is signaled by the reception of a congestion notification followed by load reduction in the network, each flow selects two points, A and B (see Figures 5, 4). As mentioned, each time a flow receives an ACK, it measures the corresponding RTT, Throughput and cwnd. In our evaluation, point A is the middle sample of all its measurements, and B is the penultimate sample during that epoch. We then estimate:

- How close the flow operates to its fair-share, according to the throughput analysis presented in section 3.
- The congestion window value that allows it to work at its fair-share, according to the Fair-Share rule (see section 4).
- The congestion window value that guarantees fair allocation of resources among the competing flows, according to the Fairness rule (see Section 4).

Note that at the evaluation for the *Fairness* rule, each flow uses the *Fairness* rule as an enhancement of the *Fair-Share* rule; each flow implements the Fairness rule at the reception of a congestion signal and the change of the fair-share, or the start of transmission. More particularly, each flow at the end of the i^{th} epoch compares the estimated fair-share with the fair-share at $(i-1)^{th}$ epoch. If the value of fair-share has changed, the flow adjusts its cwnd according to the Fairness rule, otherwise it follows the Fair-Share rule.

To study how system converges to equilibrium and how it reacts to sudden changes of traffic, we implement a contention increase scenario where we study each algorithm under the same dynamic network conditions. More particularly, we use a single bottleneck link (see Fig.8) with a bandwidth of 10Mbps where 2 flows enter the system a time difference of 0.1secs, while two more flows enter at the 10th and 20th sec of the experiment, respectively.

We measure system behavior in the presence of the proposed algorithms by monitoring measurements and cwnd values. We also use the *short-term fairness* index to capture potential bandwidth tradeoff among flows. More particularly, the *ShortTermFairness* index shows how resources are allocated among flows within short time slots:

$$ShortTermFairness = E_t\{\frac{\left(\sum_{i=1}^{n} Throughput_i\right)^2}{n\sum_{i=1}^{n}(Throughput_i)^2}\} \tag{14}$$

where, $Throughput_i$ is the throughput of the $\mathtt{i}^{\mathtt{th}}$ flow, defined as:

$$Throughput_i = \frac{TotalDataSent}{TransmissionTime} \tag{15}$$

where `TotalDataSent` is equal to the sum of original data, retransmitted data and packets' header size (in Bytes), during its connection (`TransmissionTime`).

ShortTermFairness index is sampled in short time scales and provides a measure of fairness with better granularity.

6.1. Simulation Setup

In our evaluation scenarios, we needed a mechanism at the router that would notify all competing flows simultaneously about the congestion. This is of crucial importantance, since our system model described in section 2 is characterized by a centralized feedback model, where all flows become aware of congestion events synchronously.

`DropTail` is a well-known mechanism that results in simultaneous notification of flows, due to multiple packet losses when the buffer capacity is exhausted. However, as the number of flows increases only some of the flows are notified. Consequently `DropTail` is not the appropriate mechanism for our evaluation scenarios.

`EGCN`, which is described in detail in [9] is an active queue management, which results in system-wide synchronous notification when congestion is about to occur. `EGCN` detects the condition of the link, based on: (i) the absolute value of the current average queue size and (ii) the variation of the average queue size. When the load in the network increases and buffer overflow is expected, `EGCN` marks the `ECN` bit [7], [8] in the `IP` header of the incoming packets. For these reason, we chose and implemented `EGCN` algorithm at the routers.

In our simulations, the parameters of `EGCN` were set according to [9]. That is, $\mathtt{min_{th}}$ and $\mathtt{max_{th}}$ are set to 1/4 and 5/16 of the buffer size, respectively. The queue buffer size is set based on the Bandwidth Delay product.

6.2. Throughput analysis

We implement the contention increase experiment discussed above. Each flow implements `TCP`-Reno and records its Throughput and `cwnd` samples. The relevant measurements of each competing flow are illustrated in Figures 9, 10, 11 and confirm the validity of the Throughput analysis, which we presented in Section 3. When a flow operates below its fair-share, for instance $flow_2$ during the 2nd and the 4rth sec (see Figure 9(a)), $throughput_B -$

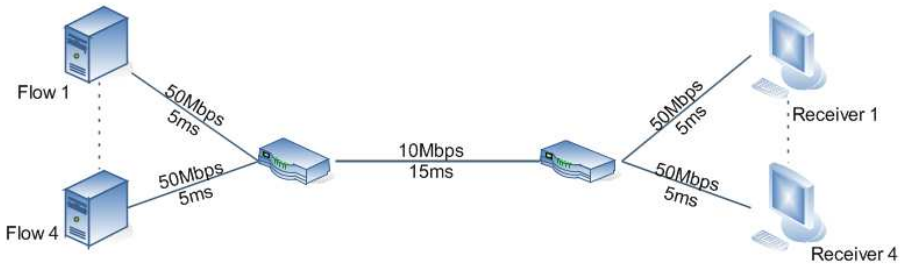
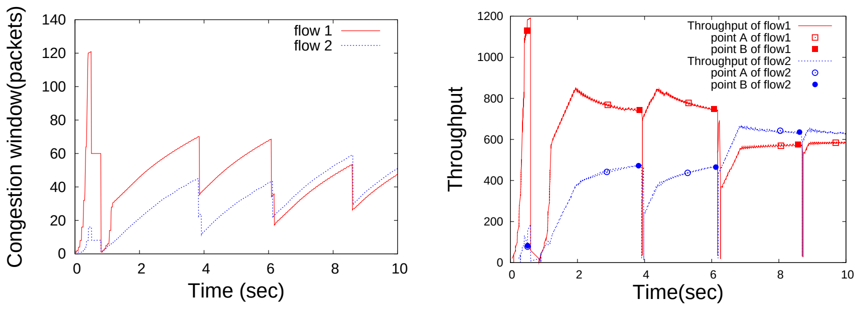


Figure 8: Simulation Topology

$throughput_A > 0$ and consequently the throughput slope is increasing (Figure 9(b)). In contrast, when a flow consumes more resources than its fair-share ($flow_1$ during the 2nd and the 4rth sec), $throughput_B - throughput_A < 0$ and the throughput slope is decreasing. Finally, when a flow operates at its fair-share, (see Figure 11(a)) during the 13th and 15th secs, $throughput_B - throughput_A = 0$ and the throughput slope is zero (see Figure 11(b)).

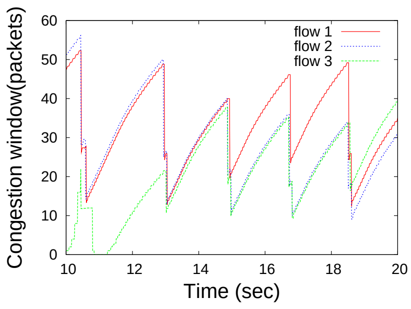


(a) Congestion window (b) Throughput

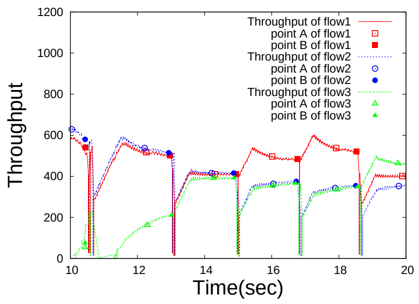
Figure 9: System samples during the 0-10sec (AIMD)

6.3. Fair-share rule

We repeat the same experiment, albeit, this time each flow implements the Fair-Share rule upon the reception of a congestion signal. Figures 12, 13 and 14 show that at the end of an epoch, where all co-existing flows measure their Throughput samples, the flows succeed in adjusting their window to their fair-share. This is confirmed by the corresponding Throughput slope

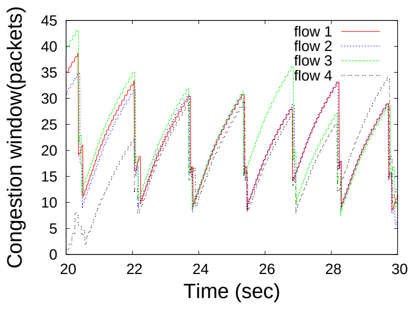


(a) Congestion window

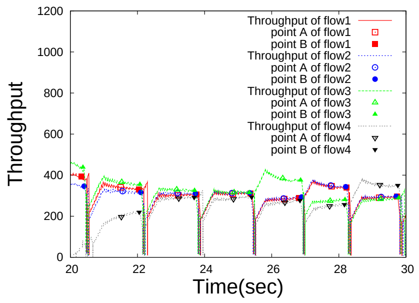


(b) Throughput

Figure 10: System samples during the 10-20sec (AIMD)



(a) Congestion window

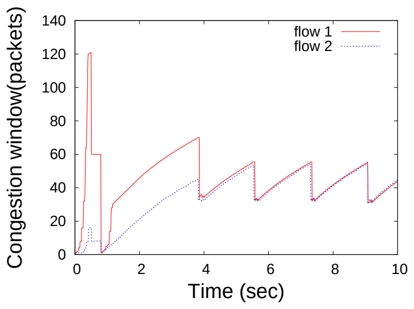


(b) Throughput

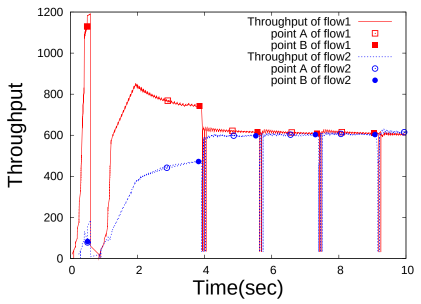
Figure 11: System samples during the 20-30sec (AIMD)

in Fig.12(b), 13(b) and 14(b), which is zero. Even when new flows enter the system, all flows, after a sequence of epoch-long measurements, adjust their rates to the new fair-share (see Figures13(a) and 14(a)).

Finally, Figure 15 confirms that the fair allocation of resources is achieved in conjunction with maintaining high levels of link utilization; system operates between the *knee* and *cliff* points, and utilization is affected momentarily (no more than an epoch), which is the time required for all flows to evaluate their new fair-share. However, although the system converges to the equilibrium state, system fairness is not achieved, since flows do not consume equal amounts of resources during their co-existing time. This is depicted by the *ShortTermFairness* index, which is sampled every 10 seconds and has the values: 0.969, 0.962 and 0.992 for the time periods 0-10secs, 10-20secs and 20-30secs, respectively.

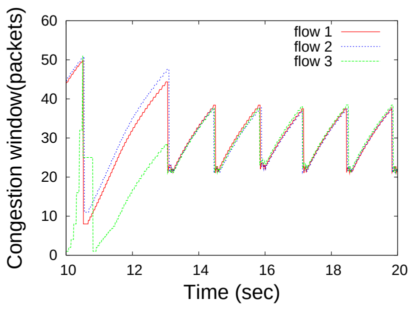


(a) Congestion window

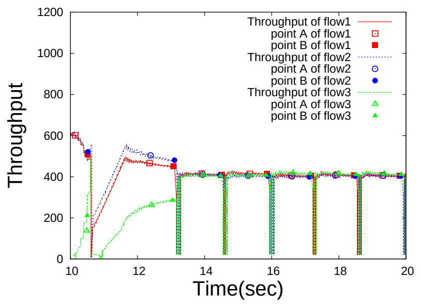


(b) Throughput

Figure 12: System samples during the 0-10sec (Fair-share rule)

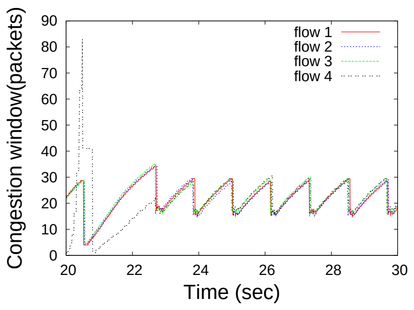


(a) Congestion window

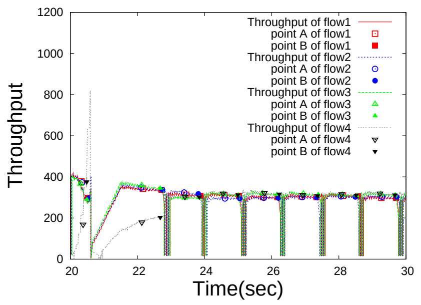


(b) Throughput

Figure 13: System samples during the 10-20sec (Fair-share rule)



(a) Congestion window



(b) Throughput

Figure 14: System samples during the 20-30sec (Fair-share rule)

6.4. Fairness rule

We repeat the same experiment, to evaluate the Fairness rule. Figures 16, 13 and 14 show how Fairness rule improves system fairness. For instance,

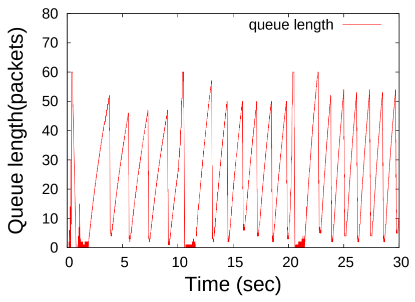
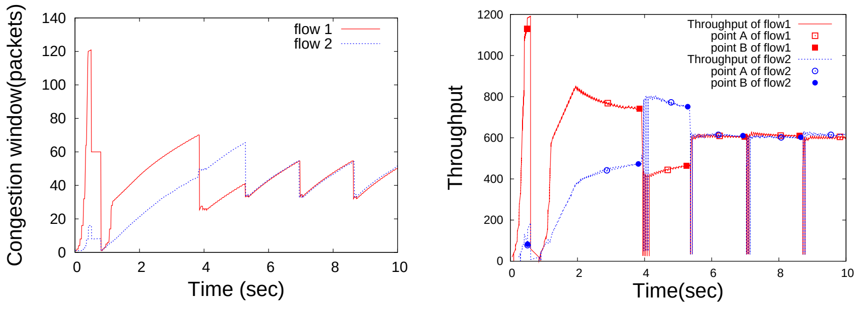


Figure 15: Queue length (Fair-share rule)

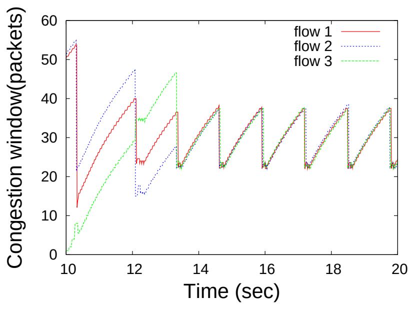
Flow 1, 2, 3 and 4 during the 22.5 and 23.8 seconds of the experiment (see Figure 18), estimate their fair-share and compare it with the fair-share value estimated during the previous epoch (during 20.5 and 22.5 seconds). Since their fair-share has changed, they apply the Fairness rule. The `cwnd` of flows 1, 3 and 4 is adjusted below their fair-share at the 22.5th sec, since they have consumed more resources during the 20.5 and 22.5 seconds (see Fig 18(b)), while $flow_4$ increases its `cwnd` beyond its fair-share at the 22.5th sec, since it was treated unfairly during the 20.5 and 22.5 seconds. At the 23.8th sec of the experiment, when the end of an epoch is signaled again, each flow applies the Fair-Share rule and operate at its fair-share.

The *ShortTermFairness* values for the time periods 0-10secs, 10-20secs and 20-30secs, are 0.994, 0.998 and 0.999, respectively. Consequently, fair allocation of resources among flows is achieved, maintaining the high utilization at the link(see Fig. 15).

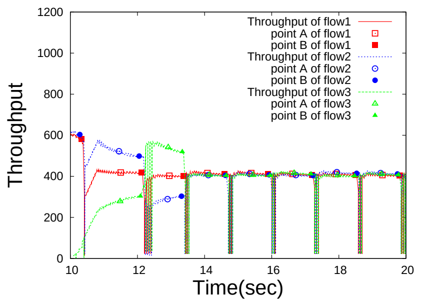


(a) Congestion window (b) Throughput

Figure 16: System samples during the 0-10sec (Fairness rule)

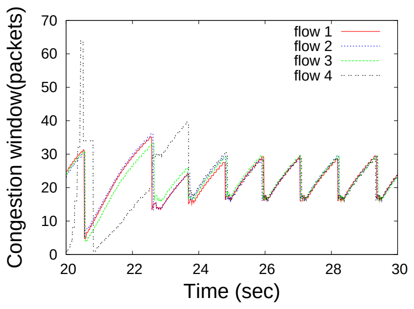


(a) Congestion window

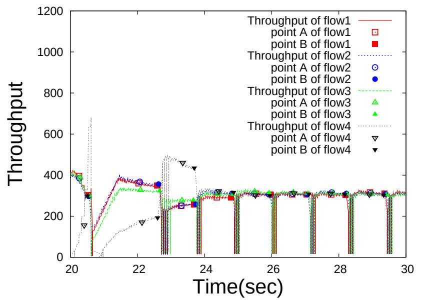


(b) Throughput

Figure 17: System samples during the 10-20sec (Fairness rule)



(a) Congestion window



(b) Throughput

Figure 18: System samples during the 20-30sec (Fairness rule)

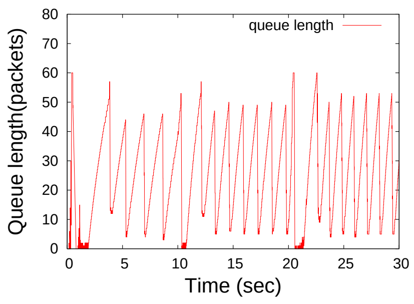


Figure 19: Queue length (Fairness rule)

7. Conclusion

In the context of the present work, we presented and evaluated a new throughput analysis that allows each flow independently to estimate efficiently its deviation from the fair-share. Based on the proposed throughput analysis, we designed new rules for congestion window adjustment. More particularly, we first showed analytically that the competing flows can adjust their operation to a dynamically-changing fair-share, based on the "Fair-Share rule and adjust further to a "lifetime" fairness by calculating resource credit, based on the "Fairness" rule. We evaluated the proposed algorithms experimentally as well, and found that our design principles match well our simulation results.

Our next research step is to incorporate the analysis and algorithms into a new end-to-end protocol specifically designed to regulate congestion based on the result of this study.

References

[1] ns 2. the network simulator - ns - 2, http://www.isi.edu/nsnam/ns/.

[2] D.-M. Chiu and R. Jain. Analysis of the increase and decrease algorithms for congestion avoidance in computer networks. *Comput. Netw. ISDN Syst.*, 17(1):1–14, 1989.

[3] Van Jacobson and Michael J. Karels. Congestion avoidance and control, 1988.

[4] V. Paxson M. Allman and W. Stevens. Tcp congestion control. *RFC2581.*

[5] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose. Modeling tcp throughput: A simple model and its empirical validation, 1998.

[6] J. Postel. Transmission control protocol. In *RFC 793*, 1981.

[7] K. K. Ramakrishnan, Sally Floyd, David Black, and Group K. Ramakrishnan. The addition of explicit congestion notification (ecn) to ip, 2001.

[8] K. K. Ramakrishnan and R. Jain. A binary feedback scheme for congestion avoidance in computer networks. *ACM Trans. Comput. Syst.*, 8(2):158–181, May 1990.

[9] A. Tsioliaridou and V. Tsaoussidis. Fast convergence to network fairness. *Technical Report: TR-DUTH-EE-2009-18, submitted.*

Ageliki Tsioliaridou received her Diploma of Electrical and Computer Engineering from Democritus University of Thrace, Greece in 2005. She is currently a PhD candidate in Electrical and Computer Engineering in the same institution. Her research interests include congestion control in packet networks, smooth data transmission algorithms and AQM based algorithms. She participated in the Organizing Committee of WWIC 2005.

Chi Zhang is a Senior Kernel Engineer at Juniper Networks. He was an Assistant Professor of Computer Science at Florida International University from 2003 to 2006. He received the B.E. degree in Electronic Engineering from Shanghai Jiao Tong University, China, in 1996, and the Ph.D. degree in Computer Science from Northeastern University, Boston, MA, in 2003. Dr. Zhang's research interests lie in the areas of network protocols, congestion control, mobile computing and QoS. He has published 28 papers and issued 1 US patent. He served on a number of program committees of networking conferences, and is on the editorial board of the Journal of Internet Engineering. Dr. Zhang received the runner-up award in the 7th IEEE Symposium on Computers and Communications (ISCC 2002), and is a member of the Phi Kappa Phi Honor Society.

Vassilis Tsaoussidis has a B.Sc in Applied Mathematics from Aristotle University, Greece and a Ph.D in Computer Networks from Humboldt University, Berlin, Germany (1995). He held appointments, and a faculty appointment at the Computer Science Department of SUNY Stony Brook (until 2000) and at the college of Computer Science of Northeastern University, Boston (until 2003). He returned to Greece in May 2003 to join the Faculty of the Department of Electrical and Computer Engineering of Democritus University, where he is now full Professor. He also held appointments as Visiting Professor at TU Berlin and MIT Boston.

Prof Tsaoussidis was/is editor for IEEE Transactions on Mobile Computing, the Journal of Computer Networks the Journal of Wireless Communications and Mobile Computing the Journal of Mobile Multimedia and the International Journal of Parallel, Emergent and Distributed Systems. He participates(d) in several Technical Program Committees in his area of expertise, such as INFOCOM, GLOBECOM, ICCN, ISCC, EWCN, WLN, and several others.

Vassilis graduated 5 Ph.D students and around 30 Masters.