On the Properties of System-wide Responsive Behavior

Ageliki Tsioliaridou, Vassilis Tsaoussidis
Department of Electrical and Computer Engineering,
Democritus University of Thrace, Greece
This work was funded by the Europen Commission and the project PENED 2003 of GSRT.

Abstract

High contention of flows is associated with unstable network behavior and unmanageable resource administration, i.e., convergence to equilibrium becomes a difficult task. In this paper, we propose a new approach to control congestion. In this context, we present an algorithm for active queue management, to notify simultaneously all competing flows about incipient congestion. Typically, AQM schemes notify only a small portion of participating flows about congestion events. Our notification scheme, called Global Notifier (GN), allows for an immediate and less aggressive adjustment of transmission windows with three consequent properties: (i) fairness is improved, (ii) smoothness of responsive flows is improved and (iii) system utilization is better regulated between the knee and the cliff. We detail our algorithm and the corresponding responsive behavior of participating flows, and we highlight significant performance results gathered by simulations.

1. Introduction

Most AQM algorithms, such as RED and ECN, notify flows randomly when the risk of congestion is increased. Notification relies on flows' responsive behavior; and randomness guarantees fairness. Since the system responsive behavior is only partially affected by random drops (i.e. a few flows will respond when contention increases) system-wide control is not guaranteed. Rather, system control requires a system-wide, synchronous responsive behavior, which, by and large, is feasible only when notification is global and responses are homogeneous, i.e., all flows respond to congestion signals. We propose a mechanism to control congestion based on global notification and responsive behavior.

Global notification is activated when congestion risk is detected; due to the large number of notified flows, system-wide backward adjustment corresponds to minimal share per flow. This allows for a responsive behavior with enhanced smoothness per flow. This benefit is in contrast to a system behavior, where a few flows respond sharply, while others benefit from their response. This strategy leads to significant transmission gaps for the responsive flows.

Global responsiveness has also another significant property. In particular, flows adjust their transmission behavior every time an implicit or explicit congestion signal is received; otherwise, a system-wide response for the participating flows, based on random signals, would have required much more time for all the flows to get their turn in responding to congestion. By the same token, we confirm the results of [1]. Authors in [1] prove that the unsynchronized decrease of windows may result in long-term fairness but cannot guarantee short-term fairness. Furthermore, they show that the synchronized packet drops result in throughput fluctuation and, eventually, to smoothness damage.

Synchronization and congestion events alone, are issues that require a lot of discussion and careful definitions. For example, congestion is typically defined as the situation where packets are dropped; however, congestion itself may be associated with more or less intensive packet losses, may be persistent or transient, or may be associated with burst or occasional losses. These situations correspond to varying definitions, time-wise, for what is a congestion event and what not; or otherwise, when an event starts and ends. In this paper, we simplified our scenarios, considering mainly deterministic events and time slots; at this stage of our work, we attempt to address two major issues: (i) to quantify the level of potential gain in terms of fairness, efficiency and smoothness and (ii) to determine experimentally a responsive behavior that corresponds to varying contention dynamics.

Within these guidelines, we propose a rather simple congestion avoidance and control scheme that aims at notifying a large portion of flows about incipient congestion. When incipient congestion is detected, the ECN bit of the packets that are buffered at the router is marked. Since more flows reduce their window almost simultaneously, the system converges to fair bandwidth allocation faster. Simulation results show that our algorithm notifies up to 45% of flows simultaneously and almost all participating flows in a very short period of time. Of course, the amount of notified flows depends largely on the level of contention, buffer sizes and processing capacities of the routers; however, in general, the number of notified flows is increased significantly. We show that this situation impacts short-term fairness, and in turn, system smoothness as well.

Furthermore, since more flows reduce their window smoothly, instead of a few reducing their windows sharply, the average oscillation of the system utilization can be better controlled; depending on the adjustment strategy, the system could reach fairness and still operate above the *knee*. Definitely, one has to investigate further, from the fairness perspective, the tradeoff between sharp reduction of few windows, instead of smooth reduction of more windows, and explore the thresholds where smoothness gains can be accomplished

along with fairness gains. Presently, we found experimentally that increasing and decreasing the window with a factor of 0.9 and 0.675, respectively, satisfies both objectives.

Although the implementation of the proposed congestion control mechanism calls for modifications at both the transport and network layers, minor efforts are needed, which, moreover, do not require header extensions. The ECN bit, which is used for the network congestion feedback, already exists in the IP header; and the new algorithm does not involve much computational burden. Along the same lines, deployment is also feasible: a single modification at a bottleneck router may suffice to trigger appropriate responses at the senders; a widespread use of the algorithm is not really necessitated.

The rest of the paper is organized as follows. In Section 2, we discuss related work on congestion avoidance and system smoothness. Next, in Section 3, we detail the proposed congestion control mechanism. In section 4, we present our expectations, and we justify our arguments and strategic decisions. Here, we also define our performance metrics, the parameters of simulations and the evaluation scenarios. Finally, in section 6, we conclude the paper.

2. Related Work

RED/ECN [3] algorithm was proposed to control traffic load, by discarding randomly packets prior to queue overflow. Authors in [2] highlight one weakness of RED, which cannot effectively adapt to changes of flow contention because of its fixed parameters. To alleviate this drawback, they proposed an adaptive model of RED/ECN, the ARED, which adjusts the value of max_p based on the traffic load. Although the authors do not extend their arguments to highlight the consequent impact on smoothness, based on their conclusion, we argue that this behavior of RED causes very sharp reactions when contention is low, and in turn, damages smoothness significantly. Authors in [4] propose another algorithm, the Adaptive RED, which attempts to adjust the dropping rate to network conditions. Adaptive RED attempts to adjust the max_{th} value to follow the network dynamics.

Many other related studies with RED have been conducted, such as SRED[6] and BLUE[5], which also attempt to adjust RED operations to network conditions. In another front, research aiming at increasing smoothness has been done at the transport layer. TFRC for example, has been proposed in [7] to satisfy the objective for smooth transmissions, without requiring more bandwidth than TCP. Due to its smooth variation of throughput over time, TFRC protocol responds slowly to bandwidth availability. To prevent the damage of system performance, an inherent assumption of TFRC is its co-existence with TCP. TFRC sending rate is based on a throughput equation, which allows it to compete fairly with TCP by trading responsiveness for smoothness.

The proposal by [1] is along the lines of the present work; however, the authors emphasize on the design aspects of congestion control at the transport layer only. Here, we try to exploit the collaborative impact of both transport and network layers.

3. Proposed congestion control mechanism

The proposed congestion control mechanism is consisted of two collaborating sub-mechanisms. The first one relies on an AQM algorithm at the network layer, while the other incorporates a transmission strategy at the transport layer. Note that collaboration here is a necessary condition for success; that is, each mechanism alone cannot achieve any gains. This condition, however, is not necessary for our scheme only, but for all congestion and avoidance schemes, presently. In this context, our inherent assumption of collaborative layers holds. This means that the router will (implicitly or explicitly) notify senders about incipient congestion and will expect a corresponding response. We investigate here both the notification and the response strategies.

3.1 Global Notifier at the router

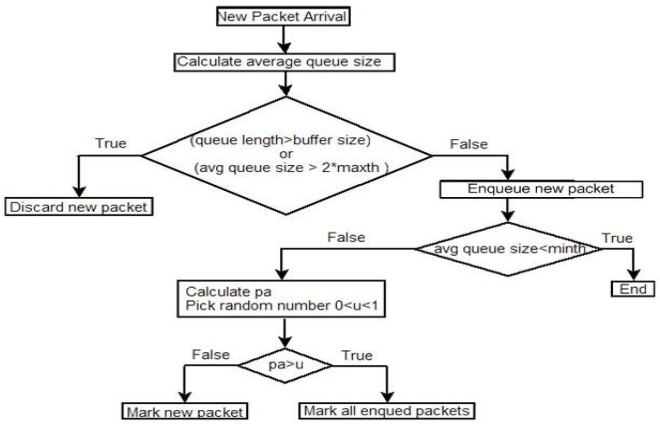

Figure 1. Flow chart of Global Notifier (GN)

The Global Notifier (GN) algorithm, which resides at the router, is briefly presented through the diagram of Figure 1. Upon arrival of each new packet, the average queue size is being estimated. The equation for the average queue size is the same as RED, that is:

If (queue is idle)

$$avg = (1 - w_q)avg + w_q avg + w_q q \quad (1)$$

else

$$avg = (1 - w_q)^m avg + w_q avg + w_q q \quad (2)$$

where, `m = current_time - q_time`

`avg` is the average queue size

`q` is the current queue length

`q_time` is last time the queue was idle

w_q is the queue weight

As soon as average queue size is calculated, a decision has to be made whether the arriving packet is going to be dropped, marked or just buffered. If the

buffer is fully utilized or the current average queue size has exceeded the 2*max_{th} the packet is being discarded; if utilization is below min_{th} the packet is simply buffered. Otherwise the packet is buffered and the probability p_a is being calculated. In particular, the corresponding equations are:

$$p_b = \max_p (avg - \min_{th}) / (\max_{th} - \min_{th}) \quad (3)$$

and

$$p_a = p_b / (2 - count \cdot p_b) \quad (4)$$

where, max_p is the maximum value for p_b

min_{th} is the minimum threshold for the avg

max_{th} is the maximum threshold for the avg

count is the number of packets since the last time the queue was idle

As long as the average queue size stays between the min_{th} and $2max_{th}$ threshold, the currently buffered packets are marked with a probability p_a. The probability p_a sets the ECN-bit to 1 not only for the just-arrived packet but also for every single unmarked packet in the queue. Furthermore, one can adjust the probability higher or lower, to increase or decrease the level of marked packets, accordingly. We do not investigate here the dynamics associated with the probability value; rather, we keep p_a as in RED and enhance the notification strategy only (i.e. mark all unmarked packets) whenever p_a leads to setting the ECN bit to 1.

Since the proposed algorithm notifies a larger number of packets (and hence flows) than RED, even with the same probability of marking, the values of max_{th}-min_{th} are reconsidered indeed. The thresholds, in fact, are used here as the regulating tool for controlling the amount of notified flows. This was deemed necessary due to the possibility of faster system response to congestion: a higher value for min_{th} could assist in avoiding link under-utilization. Experimentally, we found that a good value for the min_{th} is around 2/3 of the max_{th}.

3.2. Transmission Behavior

Since our Queue Management scheme triggers a synchronized reduction of multiple flow windows, the aggregated system response to incipient congestion is fast and, based on the typical adjustment strategies, could have been rapid, as well. This calls for a smoother window adjustment to prevent link under-utilization. Therefore, adjustments need to be smooth enough to guarantee sufficient utilization; and sharp enough to preserve system equilibrium. Note that the adjustment itself is the fairness tool of the participating flows, since, this way, the transmission gap among different windows is reduced. Based on experiments, we found that system performance can be improved by an AIMD-based multiplicative-backward and additive-upward window adjustments by a factor of 0.675 and 0.9, respectively. Noteworthy, the gentle window adjustments do not come at the expense of responsiveness, which is balanced by the quantity of responsive flows.

4. Performance Evaluation

We attempt to verify our claims based mainly on extensive simulations. We parameterize the number of competing flows; the number of flows that enter and leave the system (i.e. emulating the dynamics of contention) throughout the communication; and the diversity of RTTs of participating flows. Our scenarios allow for further analysis of (i) system performance under stable contention, (ii) measurements of responsiveness and smoothness when bandwidth becomes available or is consumed gradually or rapidly, (iii) impact of RTT diversity on system behavior and (iv) comparison of long- versus short-term fairness properties of the two AQM approaches, namely classic RED and Global Notifier.

Due to unpredictable contention dynamics of packet networks that mainly appear as a consequence of statistical multiplexing, network flows are obliged to a constant increase/decrease transmission strategy. That allows for detecting network capacity and controlling congestion, respectively. However, statistical multiplexing relies on buffering, which in turn allows for utilizing temporarily more than the link capacity. Therefore, increase/decrease is feasible without trading link utilization, provided that the operational spectrum of aggregated transmission strategies is confined within the buffer space. This space corresponds to the detectable points *knee* and *cliff*. That said, efficiency can be preserved as soon as the aggregated transmission window is regulated to operate within those limit.

An associated issue is also the impact of the aforementioned strategy (i.e., to operate between the knee and the cliff) on fairness. Fairness is achieved through backward adjustments, and convergence speed is proportional to the scale of adjustment. Naturally, aggressive downward adjustments improve fairness but damage utilization. However, there are two important issues that have to be considered: (i) the above analysis relies on a hypothesis where system response is synchronous; this is not the case with classic RED schemes and (ii) in asynchronous systems, a rapid response of a single flow cannot balance, in terms of fairness, the same total downward adjustment which is produced by smooth responses of more flows. In this context, our proposed scheme is expected to exhibit (i) better utilization due to avoiding packet drops (ii) better fairness, smoothness and responsiveness.

4.1 Performance Metrics

System `Goodput` and `Throughput` are used to measure the overall system efficiency in terms of bandwidth utilization.

`ShortTermFairness` and `WorseCaseFairness` are used to reflect how fairly are the resources consumed by users. They are defined, respectively, as:

$$ShortTermFairness = E_t \left\{ \frac{\left(\sum_{i=1}^{n} throughput(t)_i \right)^2}{n \sum_{i=1}^{n} throughput(t)_i^2} \right\} \quad (5)$$

$$WorstCaseFairness = \frac{\min_{1 \le i \le n} throughput_i}{\max_{1 \le i \le n} throughput_i} \quad (6)$$

where, $throughput_i$ is the throughput of the i^{th} flow and n the total number of flows (see also [1]).

To measure *system smoothness*, we use the `MaxAvgCwndDeviation` and the `MaxCwndDeviation`.

`MaxAvgCwndDeviation` captures the maximum gap of the average congestion window for a time duration t; while `MaxCwndDeviation` reflects the maximum gap of congestion windows that are frequently sampled at time snapshots $t_{current}$. Therefore, the `MaxCwndDeviation` metric attempts to reveal cases that may not be captured by the `MaxAvgCwndDeviation` metric. In this context, we use both a short- and a long-term metric to capture the window oscillations and system smoothness. That said, we define $t_{current}$ as the time instant, where samples are taken; and we calculate the average window and deviation at that instant. The frequency of measurements has been here determined as 0.01 sec, a time-slot scale that corresponds to 2-3 RTTs and guarantees sufficient granularity. Unlike $t_{current}$, time t is defined as the *period* from t_o to $t_{current}$.

Consequently, we define `MaxAvgCwndDeviation` and `MaxCwndDeviation`, respectively, as:

$$MaxAvgCwndDeviation = \max_{1 \le i \le n} | SystemAvgCwnd(t) - awnd_i(t) | \quad (7)$$

$$MaxCwndDeviation = \max_{1 \le i \le n} | avg\_cwnd(t_{current}) - cwnd_i(t_{current}) | \quad (8)$$

where, $awnd_i$ is the average congestion window of the i^{th} flow, $cwnd_i$ is the congestion window of the i^{th} flow, and `SystemAvgCwnd` is the system-wide average congestion window and is given by the following equation:

$$SystemAvgCwnd(t) = \frac{\sum_{i=0}^{n} awnd_i}{n} \quad (9)$$

`avg_cwnd(t)` is the average of all flows' congestion window size at time t and is defined as:

$$avg\_cwnd(t) = \frac{\sum_{i=1}^{n} cwnd_i(t_{current})}{n} \quad (10)$$

Hence, (7) captures the long-term coarse-grain smoothness, while (8) captures a fine-grained, short-term notion of smoothness.

4.2 Simulation Results

For our simulation experiments we have used the *ns* network simulator [7]. The topology of the simulated network is shown in Figure 2. A set of TCP senders (sender_1 through N) is connected with a set of TCP receivers (receiver_1 through N). The queue buffer size is set based on the *Bandwidth X Delay* product. Simulation time is fixed to 100 seconds and all senders start transmitting packets within the first 2 seconds.

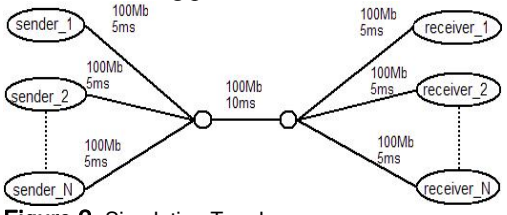

Figure 2. Simulation Topology

4.2.1 Notification efficiency

At the first stage of our experiments, we attempt to observe and quantify the comparative efficiency of our notification scheme. That is, we investigate how long it takes with RED and with GN to notify all flows about incipient congestion. We present sample – but representative[1] – results with 200 flows in Fig. 3. There, we depict the time required for system-wide notification. We see that the GN notifies all senders in the first 2 seconds, while RED needs 8 seconds. This observation justifies high expectations regarding the fairness performance of GN.

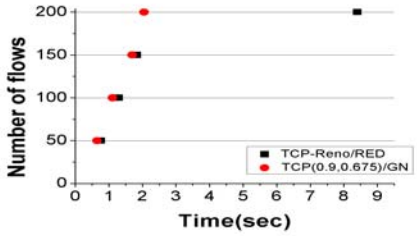

Figure 3. System Notification

4.2.2 System Utilization

In order to capture the proposed algorithm's gains in system utilization, we simulated a scenario of high contention, with the number of competing flows varying from 160 to 300.

In Figures 4 and 5 the experimental results reveal the failure of TCP-Reno, in the presence of RED, to capture link capacity. Due to unsynchronized windows increase/decrease, the system behavior is unpredictable and senders are not able to estimate the network contention. This is reflected by the fact that system throughput decrease does not correspond to Goodput increase, from a point onwards. In contrast, Goodput evolves in reverse proportion to Throughput (see Fig.4).

Next, we applied GN algorithm rather than RED. Since GN notifies almost all flows about incipient congestion, more senders reduce their window. When the number of flows is relatively small, from 160 to 220,

[1] More experiments have been conducted with varying number of flows and notification thresholds (i.e., percentages) to confirm the findings.

there is some link underutilization. This is confirmed by Fig. 4 and 5, where Throughput increase translated into Goodput increase as well. As the number of flows increases further the system reaches the link capacity.

At the last stage of this set of experiments, we adjusted the protocol's increase/decrease parameters α,β to 0.9 and 0.675, respectively[2]. We conclude this stage of experiments with two interesting conclusions: (i) GN cancels the occasional system underutilization. The reasoning behind this observation is rooted again on the dynamics of statistical multiplexing: even though the system is underutilized on the average, there are occasions of temporary buffering and RED-triggered packet drops. GN applies marking instead, and in turn, avoids unnecessary timeouts and sharp reactions. (ii) By the same token, GN achieves a better Goodput to Throughput ratio when the link is over-utilized; the modified responsive behavior of more flows works also in favor of congestion avoidance.

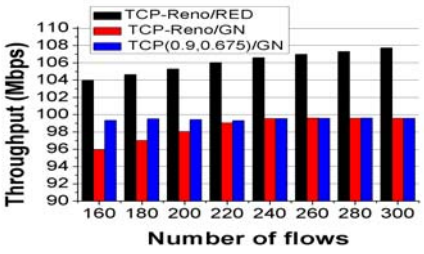

Figure 4. System Throughput

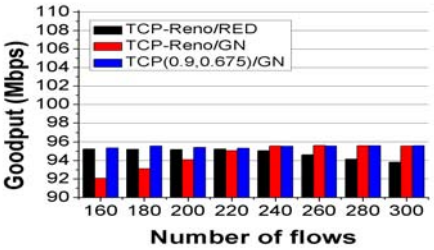

Figure 5. System Goodput

4.2.3 System Fairness

At the second stage, we concentrate on measurements of fairness. In Figures 6,7 we show the maximum and minimum values of short-term fairness. Short-term fairness is improved at the presence of TCP (0.9, 0.675) with GN compared to TCP-Reno with Red. We observe in Figure 8 that the comparative short-term fairness widens its gap after the 2nd sec, when the flows start adjusting their transmission rates in response to congestion signals.

Since the number of flows is high, a temporary unfair allocation of bandwidth for a small portion of flows cannot be captured with long-term fairness. In this context, we enhance our measurements with worst-case and allotted fairness. Figure 9 shows that worst-case fairness is improved with synchronized notifications.

[2] The values have been selected based on separate experiments, which are not reported here.

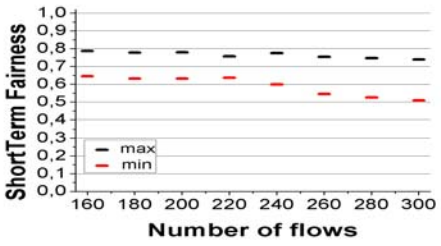

Figure 6. System short-term Fairness (with RED)

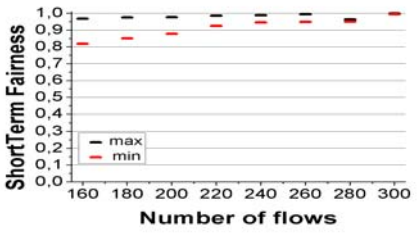

Figure 7. System Alloted Fairness (with GN)

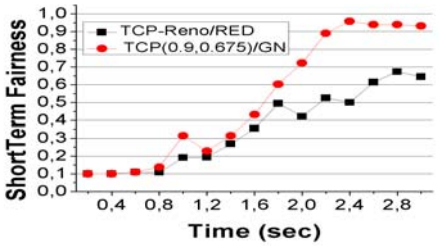

Figure 8. System Allotted Fairness

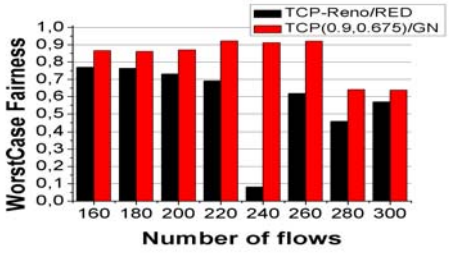

Figure 9. System WorstCase Fairness

4.2.4 System Smoothness

Next, we evaluated the algorithm's potential for smoothness. For this purpose, we used the metrics given by equations (7) and (8). Figures 10,11,12 and 13 depict that the proposed strategy almost doubles smoothness performance.

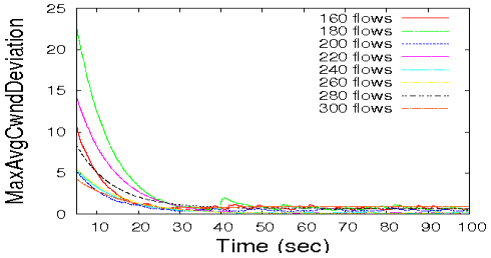

Figure 10. MaxAvgCwndDeviation (with RED)

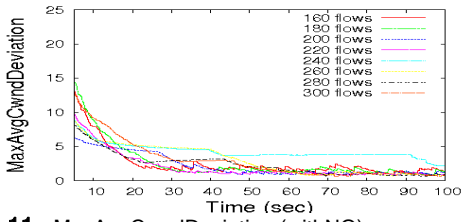

Figure 11. MaxAvgCwndDeviation (withNG)

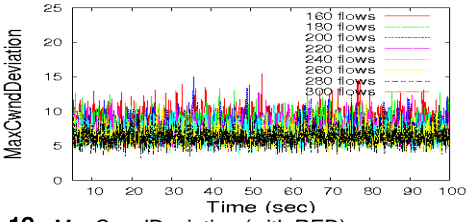

Figure 12. MaxCwndDeviation (with RED)

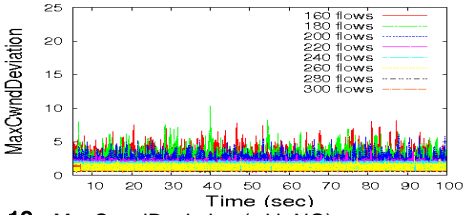

Figure 13. MaxCwndDeviation (with NG)

4.2.5 System Responsiveness

To evaluate system responsiveness to bandwidth availability, we simulated a scenario where, during the simulation time, some flows finish their work earlier. In particular, 300 flows initially compete for the same link until some of them (30, 60, 90, 120 and 150 flows) leave the channel. Figures 14 and 15 depict the corresponding results.

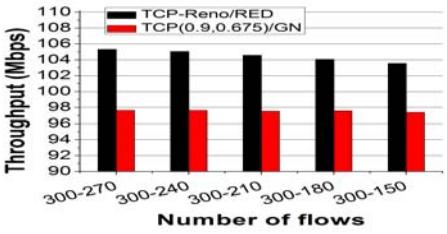

Figure 14. System Throughput

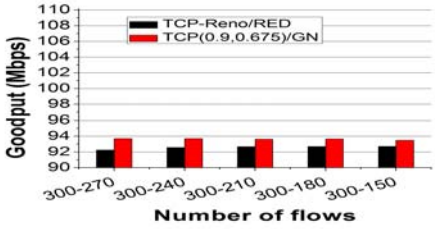

Figure 15. System Goodput

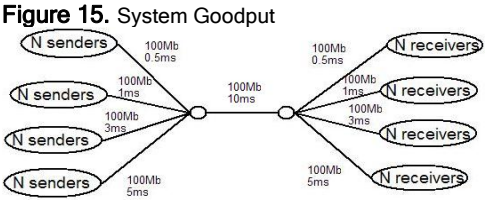

Figure 16. Simulation Topology

4.2.6 Impact of diverse RTTs

Next, we simulate a scenario (Figure16) with different RTTs, in order to uncover any potential undesirable property of GN due to RTT diversity. The simulation results (Figures 17, 18) show, however, that similar comparative performance is exhibited here as well.

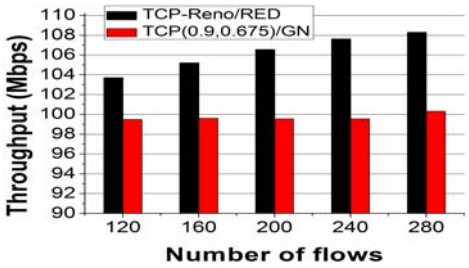

Figure 17. System Throughput

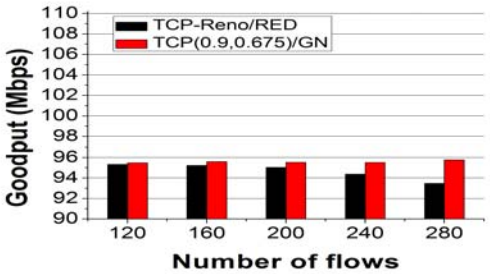

Figure 18. System Goodput

5. Conclusions

We have presented results to support our arguments that system-wide notification allows for better system smoothness. An interesting result relates with the trade-off between fairness and smoothness: in contrast to traditional theories, we show that, in the context of Global Notifier, smoothness and fairness do not follow contradicting dynamics.

References

[1] Chi Zhang, Vassilios Tsaoussidis, "Improving TCP smoothness by synchronized and measurement-based congestion avoidance", ACM NOSSDAV, 2003.

[2] Li Lei, Pan Yong and Shi Hongbao, "An Adaptive Model of RED/ECN Parameters", IEEE/Communication Technology Proceedings, WCC-JCCT,2000.

[3] Sally Floyd and Van Jacobson, "Random Early Detection Gateways for Congestion Avoidance", IEEE/ACM Transactions on Networking, 1993.

[4] S. Floyd, R. Gummadi, S. Shenker, "Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management", 2001.

[5] W. Feng, D. Kandlur, D. Saha, K. Shin, "Blue: A New Class of Active Queue Management Algorithms", UM CSE-TR-387-99, 1999.

[6] T. J. Ott, T. V. Lakshman, and L. H. Wong, "SRED: Stabilized RED", INFOCOM, 1999.

[7] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-Based Congestion Control for Unicast Applications", In Proc. of ACM SIGCOMM , 2000.