How to avoid TCP Congestion without dropping Packets: an Effective AQM called PINK

Maurizio Casoni[a], Carlo Augusto Grazia[a], Martin Klapez[a], Natale Patriciello[a]

[a] *Department of Engineering Enzo Ferrari*
University of Modena and Reggio Emilia
via Pietro Vivarelli 10, 41125 Modena, Italy.

Abstract

This paper proposes PINK (Passive INverse feedbacK), a queue management algorithm designed to indirectly impose a certain resource allocation policy on defined sets of client hosts. PINK adds intelligence at intermediate nodes that connect client hosts to bottleneck links or to external networks in general, allowing these nodes to dynamically modify the TCP Acknowledgements (ACKs) segments passing through. The modification consists in replacing advertised Receive Window fields (RCV.WNDs) with custom values, in order to enforce a specific bandwidth utilization upper bound. To compute new RCV.WND values, PINK needs only the number of active connections, the flows RTTs and the transmission channel bandwidth. It follows that PINK permits to impose a centralized bandwidth management without the cooperation of clients, which means that no modification or addition to end hosts is needed. Furthermor e, as demonstrated in this paper, our proposal does not constrain client hosts performance without purpose; on the contrary, PINK improves efficiency on multiplexed channels by exploiting their capacity and by maintaining a low queuing delay, and guarantees optimal flow fairness without forcing any packet drop. We validate PINK performance in multiple scenarios by using the ns-3 network simulator.

Email addresses: `maurizio.casoni@unimore.it` (Maurizio Casoni), `carloaugusto.grazia@unimore.it` (Carlo Augusto Grazia), `martin.klapez@unimore.it` (Martin Klapez), `natale.patriciello@unimore.it` (Natale Patriciello)

Keywords: AQM, TCP, fairness, rate control, receiver advertised window

1. Introduction

In recent years, due to the explosion in the number of connected devices, the interest on congestion issues over computer networks has risen as well. In particular, researchers and developers have implemented and tested various proposals, often with the common property of not being limited to a particular layer of the network stack, but instead collaborating over different layers to resolve a particular problem. Moreover, logical layers in networks are not seen anymore as static and closed black-box, but they frequently interact across-the-border.

The most sensitive transport protocol to network congestion is, without any doubt, Transmission Control Protocol (TCP): it is a stream-based, ordered and reliable protocol, and it must react to congestion to preserve its properties. The downside of recovering algorithms is a performance degradation; its degree depends on the intensity of the congestion.

Active Queue Management (AQM) algorithms and packet schedulers are the most representative frameworks that enforce cross-layer collaboration. For example, if network congestion is detected among TCP flows by a router, its AQM algorithms could decide to selectively drop packets, in order to indirectly signal senders to slow down the transmission rate. Direct signalling (such as ECN, RFC 2481) has also been proposed, but this comes at the cost of updating all nodes across the networks, together with senders and receivers.

In this paper we present PINK (Passive INverse feedbacK), an algorithm that has been designed to run on access gateways as an AQM technique that prevents congestion. The novelty of PINK is encased in its ability to fairly divide a bottleneck bandwidth between TCP flows, in order to transparently impose a fair rate and mitigate congestion without a single packet drop. This allows PINK to be independent from the TCP congestion control algorithm employed by the end users, to be independent from the RTT of each flow and to be extremely efficient in terms of latency and jitter. PINK operates by

modifying the ACK packets, without adding non-standard headers or options to them and without requiring any modification on any other nodes along the path, end nodes included. Moreover, to maintain the standard semantic of TCP algorithms in terms of channel exploitation, PINK is equipped with an algorithm that reassigns, if necessary, the resources when one or more flows are not able to exploit their optimal rate as provided by PINK.

The discussion is organized as follows: Section 2 summarizes related work. Section 3 describes the PINK algorithm. Section 4 presents a mathematical analysis of the gateway queue occupancy upper bound when employing our proposal. Section 5 introduces our validation scenarios, while Section 6 shows the simulations results. In Section 7 the conclusions of our work are drawn.

2. Related Work

The AQM world is still an active topic in the literature, in this section we carefully introduce the more recent and relevant works together with feedback-based congestion control algorithms, which will take part to the experimental evaluation.

The AQM problem is strongly coupled to the TCP family and also sound and detailed analysis have been provided such as [1], AQM algorithms in fact basically try to exploit some smart solutions in the middle nodes between the end-to-end communication point of a TCP connection. The initial approach to let TCP cooperate with routers is the Explicit Congestion Notification (ECN) presented in [2]. The authors proposed an algorithm which extracts the network status from successive binary congestion information and estimate a fair window size. With such a system, the window size reduction is identical to that caused by the fast retransmit mechanism of TCP Reno and does not require dropping any packet. From one side, ECN is effective in reducing packet loss while, on the other, the binary feedback alone is not enough to avoid window and network behaviour oscillations, and it does not allow to operate fine-grained adjustments.

One of the first attempts to explicitly control network congestion through a

more substantial feedback has been proposed by Gerla et al. in [3]. The authors proposed a Generalized Window Advertising (GWA) for TCP congestion control algorithms that aims to avoid window oscillations and the related fluctuations in offered load and network performance. The key idea of GWA is to operate on the TCP advertised Receive Window (RCV.WND) set by the receiver in the header of TCP ACK segments in order to convey both the receiver available buffer space and the network congestion status. Unfortunately, to do this, GWA requires a modification of the existing protocol stack of the end-nodes, carrying congestion information for the GWA-TCP sender at the IP level.

A similar approach is described in [4, 5]. This work also proposes the use of the RCV.WND field in TCP headers to convey network congestion notifications. Even if the congestion control algorithm as well as the protocol implementation are different from GWA, the drawbacks are the same; in fact, modifications to the end hosts stacks are required, resulting in a not transparent solution. Moreover, both previous works are based on the router queue length, which could be an outdated congestion information when networks with high propagation delay (like satellite ones) are in place, causing instability. The problem of the stability of congestion control for networks with large round-trip communication delays has been considered in [6] to steer the design of the IMC-PID congestion controller.

A very promising solution proposed in [7] shares proposal goals with our work. The paper in fact focuses on the fairness and QoS provisioning in a mixed environment, e.g. with multiple variants of TCP congestion controls, by migrating the control rate from the end applications to a general protocol in the middle. In fact the authors proposed a Rate Management Protocol (RMP) that controls the rate of all flows (in a router) in order to accommodate QoS provisioning. The RMP algorithm in the middle is completed with a novel TCP congestion control algorithm on the end nodes (called TCP-RMP congestion control) that is able to infer the fair rate for each flow when coupled with RMP. The clever mix of this two solutions is that RPM protocol helps to policing "aggressive" flows (UDP and aggressive TCP versions) while TCP-RMP congestion

control enables "polite" to achieve the fairness point. This problem of multi-
90 variety of traffics and protocol over a networks has been investigated also in [8] to devise the IAPI algorithm, a stable and robust AQM tested under various scenarios including cases involving different types of background traffic and a case involving a multiple bottlenecks.

Another AQM algorithm called GREEN [9] is not so far from our proposal
95 due to its rate-based approach; GREEN operates giving each packet a drop probability that depends on the flow RTT, on the number of active flows, and on the Maximum Segment Size (MSS) encountered during a sampling time. However, GREEN exhibits some drawbacks: (i) it does not have a reference implementation; (ii) it infers the MSS by considering the highest value only, which
100 results in a coarse-grained drop policy; (iii) the RTT measurement assumes the usage of TCP options (i.e. it leaves to end nodes the responsibility of providing RTT values to the gateway). GREEN is therefore a non-transparent solution for end nodes too. The rate-based feature has become very important in the last years; following this approach also solutions for achieve a fine-grained conges-
105 tion control for multicast applications has been investigated in [10]. Moreover, the rate-based and the queue-based approach have been coupled together in the RaQ algorithm [11], a dual loop feedback control algorithm that takes the input rate and current queue length to calculate the packet dropping/marking probability. Thus, the rate feedback control enables RaQ to respond to congestion
110 rapidly, in this way it can decrease the packet loss due to the buffer overflow. At the same time the queue length feedback control stabilizes RaQ's queue length around given target, so it can be tuned to achieve the desired queueing delay and mitigate delay jitter.

In an another work, more related to the packet scheduler world instead of the
115 AQM one, a solution to harmonize the bandwidth sharing between several TCP flows competing for the same bottleneck link is studied [12]. This interesting approach proposes to delay the ACKs in order to force each TCP sender to get a specific yet fair amount of bandwidth at the bottleneck link. The only drawback of this solution is that it can only act on the bandwidth by artificially increasing

the RTT, which is something likely undesirable in many real-time applications that use TCP.

To address these issues on satellite-based emergency networks that employ high delay links we present in [13] a preliminary work on PINK. We designed our proposal on the basis of this last work, by also cross-comparing our contribution with the aforementioned systems, in order to validate it and provide an answer to the drawbacks that are being exposed in this Section. We based our comparison testbed on recent works such as [14] and [15], which are also based on AQM techniques. In particular, the latter proposes a comparison with recent AQM implementations like CoDel and PIE with a decade old variant of RED called ARED. The interesting result of this work is that, in several instances, ARED obtains better results. In [16] it is shown instead that while AQM algorithms are able to significantly improve performance on the long run they also exacerbate TCP flows unfairness, especially among TCP flows with different RTTs, and may lead to large latency spikes caused by queuing delays when flows startup.

3. The PINK algorithm

In this section, the PINK algorithm is described through both a theoretical explanation and implementation directives.

PINK is a per-flow AQM algorithm, designed to be deployed on a gateway that provides network access to a set of client hosts. It exploits the flow control of TCP, and in particular the one of the receiver, through the well-known field "Window size" (RCV.WND) of the TCP header. This field represents the quantity of bytes that the destination is currently willing to receive, and an RFC-compliant sender will not exceed this value when transmitting segments, choosing as its window size the minimum value between congestion and receiver window.

Basically, PINK computes (for each ACK packet) the value for the advertised RCV.WND, which will be then written on the header. The key concept is that this value is calculated so as to prevent bottleneck congestion, tricking the

sender.

Mathematically, such value equals to:

$$RCV.WND_i^{pink} = B_i = \left\lfloor \frac{BW \cdot RTT_i^{min} \cdot c}{n} \right\rfloor \quad (1)$$

Considering that $\frac{BW}{n}$ is the bandwidth allocation given to each active flow i, in Equation 1 BW is the bottleneck bandwidth, n is the current number of active flows, RTT_i^{min} is the minimum RTT calculated by PINK for the flow i while $RCV.WND_i$ is the new RCV.WND value to write in the TCP window field, equal to B_i that represents the burst value of flow i. The constant parameter c, called the "exploitation parameter", ranges from 0 to 1 and represents the channel exploitation factor. Is has been introduced to allow the network designer to tune the amount of bandwidth shared among nodes, in order to compensate packet overhead for headers introduced by network layers. Indeed, when $c = 1$ the entire bandwidth is considered to be used by TCP for the application data, leaving no space for TCP, IP, and link layer headers. This exceeding amount leads to the use of some queue space (that therefore generates a low degree of congestion) that can be avoided by using $c < 1$. The bottleneck bandwidth is assumed to be an offline and always available information; the number of active connections, instead, is actively tracked by counting the number of SYN, FIN and RST packets that are received or sent. It is likewise possible to efficiently calculate the RTT of each flow [17, 18, 19].

PINK operates per-flow since it divides the bottleneck bandwidth over the number of active flows on the channel [1]. We reported in Figure 1 the operational scheme of PINK, where *pinkwnd* represents the calculated RCV.WND value.

As a matter of fact, TCP connections are bi-directional, because data could be exchanged both ways (upload and download). In concrete terms, there are

[1] Is important to note that PINK operates on the reverse path, hijacking the ACK packets, and a possible cause of congestion like the entrance of a new flow is notified to the sender without waiting the entire RTT flow but instead is notified through the first ACK available resulting in a notification as fast as the Backward Congestion Notification [20].

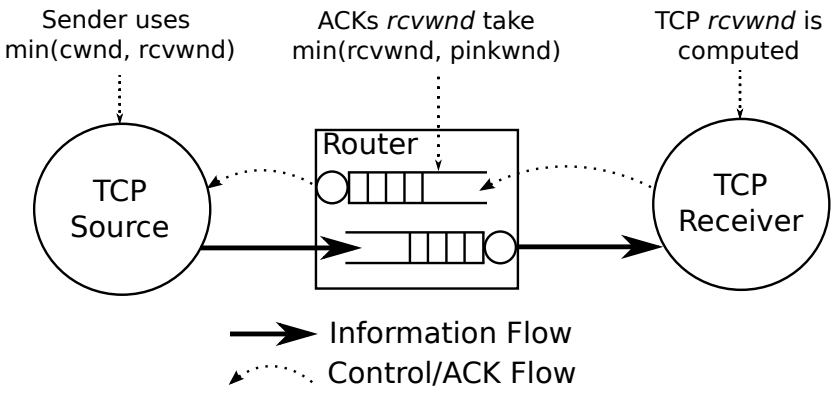

Figure 1: PINK algorithm on a simple network.

two information flows and two ACK flows, but for the sake of brevity in this paper we only investigate the uplink direction, applying PINK on the ACKs sent by remote receiver nodes. The setup could be flawlessly extended to prevent congestion on both directions, by applying PINK on the ACKs sent by the sender too.

PINK is basically designed for access networks, in these environments the bottleneck data rate allows PINK to compute the TCP Checksum after the alteration of the TCP advertised Receive Window, (an operation that must be done for each and only ACK that needs to be changed) without introducing a performance overhead. This means that PINK can be installed on a general access-network node through a simple software update. The situation is different if PINK wants to be used on a more than Gigabit-level backbone network in which to avoid performance degradation is mandatory to compute hash functions through specialized hardware [21], instead of performing these operations in software as it is feasible in lower-bandwidth access systems.

3.1. PINK resource reallocation

We created an algorithm to prevent a waste of resource for PINK adapted from a previous work [22]. In particular we considered what could happen in a multi bottleneck environment in which PINK operates on different nodes/routers. It may happen that, for one or more particular flow, the $RCV.WND_i^{pink}$ calcu-

Table 1: Variables of the PINK algorithm for resource reallocation

n	number of active flows
n_{bad}	number of bad flows
n_{good}	number of good flows
s_i	state of the i-th flow (*good* or *bad*)
BW	bottleneck bandwidth
BW^{free}	total wasted bandwidth
BW_i^{free}	wasted bandwidth by i-th flow
RTT_i^{min}	minimum RTT of the i-th flow
$RCV.WND_i^{current}$	current RCV.WND of the i-th flow ACKs
$RCV.WND_i^{pink}$	RCV.WND computed by PINK for the i-th flow ACKs

lated by PINK is higher then the current receiver windows of the ACK packet of the i-th flow, hereafter called $RCV.WND_i^{current}$. Because of it is semantically wrong to increase the RCV.WND of an ACK packets in the middle of the path, the PINK algorithm operating in the node under discussion is missing a resource equal to $\frac{RCV.WND_i^{pink} - RCV.WND_i^{current}}{RTT_i^{min}}$, that is the amount of bandwidth not used by the i-th flow.

This situation is captured through the algorithm described here with the help of the variables described in Table 1:

$$RCV.WND_i^{pink} = \left\lfloor \frac{BW \cdot RTT_i^{min} \cdot c}{n} \right\rfloor + \left\lfloor \frac{BW^{free} \cdot RTT_i^{min} \cdot c}{n_{good}} \right\rfloor \quad (2)$$

which is equal to Equation 1 by default because BW^{free} is initialized as zero.

$RCV.WND_i^{current} < RCV.WND_i^{pink}$ *and* s_i *is* good. The i-th flow is unable to use all the assigned bandwidth; its state moves therefore from *good* to *bad* and the unused bandwidth is reassigned to other *good* flows.

$RCV.WND_i^{current} < RCV.WND_i^{pink}$ *and* s_i *is* bad. The i-th flow is unable to use all the assigned bandwidth; its state remains *bad* and the unused bandwidth is reassigned to other *good* flows.

$$n_{bad} \leftarrow n_{bad} + 1$$
$$n_{good} \leftarrow n_{good} - 1$$
$$BW^{free} \leftarrow BW^{free} - BW_i^{free}$$
$$BW_i^{free} \leftarrow \frac{RCV.WND_i^{pink} - RCV.WND_i^{current}}{RTT_i^{min}}$$
$$BW^{free} \leftarrow BW^{free} + BW_i^{free}$$

Figure 2: Flow i moves from *good* to *bad* state

$$BW^{free} \leftarrow BW^{free} - BW_i^{free}$$
$$BW_i^{free} \leftarrow \frac{RCV.WND_i^{pink} - RCV.WND_i^{current}}{RTT_i^{min}}$$
$$BW^{free} \leftarrow BW^{free} + BW_i^{free}$$

Figure 3: Flow i remains *bad*

$RCV.WND_i^{current} > RCV.WND_i^{pink}$ *and* s_i *is* bad. The i-th flow is able to use all the assigned bandwidth; its state moves from *bad* to *good* and the unused bandwidth is removed.

$$n_{bad} \leftarrow n_{bad} - 1$$
$$n_{good} \leftarrow n_{good} + 1$$
$$BW^{free} \leftarrow BW^{free} - BW_i^{free}$$

Figure 4: Flow i moves from *bad* to *good* state

4. PINK guarantees

In this section we analyse the impact of PINK on the gateway queue level, considering in particular the worst-case upper bound value of queue occupancy.

Theorem 1. *Let $Q_{gw}(t)$ be the output queue length of the gateway at the generic time t and let RTT_{net} be the average round trip time of the network. If the link bandwidth is constant and equal to BW, with PINK as AQM algorithm the following inequality holds for any time t:*

$$Q_{gw}(t) \leq BW \cdot RTT_{net}. \tag{3}$$

PROOF. Consider a time interval with a constant number of n active TCP flows. Consider now a generic i-th TCP flow; it will send in the network a burst B_i which is at most equal to the product $BW_i \cdot RTT_i^{min}$ each RTT_i, as showed in Equation 1 with $c = 1$ for a worst-case analysis. Consider the case in which all flows send their bursts simultaneously, at the same time t. The gateway queue, immediately before t, can be either empty or containing packets; this divides the proof in two cases.

Case 1: Empty queue. The gateway queue is filled with the aforementioned bursts and it follows that:

$$\begin{aligned} Q_{gw}(t) &= \sum_i B_i \\ &= \sum_i BW_i \cdot RTT_i^{min}. \end{aligned} \tag{4}$$

Considering RTT_i^{min} upper bounded and equal for each flow to RTT_{net}, we can write:

$$\begin{aligned} Q_{gw}(t) &= \sum_i BW_i \cdot RTT_{net} \\ &= (\sum_i BW_i) \cdot RTT_{net}. \end{aligned} \tag{5}$$

Considering Equation 1 we can easily write its fair bandwidth allocation rule in the form $BW \geq \sum_i BW_i$, and by substituting it to (5) we get the thesis.

Case 2: Non-empty queue. Let the gateway queue, immediately before t, contain at least one packet p. After receiving the aforementioned bursts it follows that:

$$\begin{aligned} Q_{gw}(t) &= p + \sum_i BW_i \cdot RTT_i^{min} \\ &> \sum_i BW_i \cdot RTT_i^{min} \\ &= BW \cdot RTT_{net}. \end{aligned} \tag{6}$$

This contradicts our assumptions. The packet p must belong to one of the active TCP flows (remember that PINK operates on an exclusive TCP queue). Consider also that the packet p belongs to the i-th flow. If follows that, at time

t, the gateway queue would contain an amount of packets belonging to the flow i which is:

$$\begin{aligned} p + BW_i \cdot RTT_i^{min} &> BW_i \cdot RTT_i^{min} \\ &= B_i. \end{aligned} \tag{7}$$

The amount of packets in the queue, belonging to the flow i, would be greater than the product $BW_i \cdot RTT_i^{min}$, which is $RCV.WND_i$; this is absurd because it violates the TCP property.

To conclude, by analysing both case 1 and case 2 we completed the proof. Theorem 1 is proved.

This way it is possible to have an upper bound guarantee about the queue length and, consequently, an upper bound on the maximum queuing delay introduced by the gateway. A key point of Theorem 1 is that this deterministic bound is equal to the standard suggested buffer size [23], which makes PINK easy to deploy, while this upper bound in the queue occupancy level prove the effectiveness of the PINK algorithm and its drop-free behaviour.

Corollary 2. *The space complexity of PINK is linear.*

Proof. In light of Theorem 1 we know that the queue occupancy of the gateway at a generic time t is upper bounded by the bandwidth-delay product. However, the used queue memory is not the only space memory allocated by the PINK queue. In fact, it also needs the list of packets waiting for RTT tracing (i.e. packets waiting to receive ACKs and, if it is the case, to update the RTT_i^{min} of a flow i, together with the state of the flow s_i and the unused bandwidth BW_i^{free} if it is the case). By Theorem 1 we know that, each RTT_{net}, the amount of data collected in the queue is deterministically bounded (Equation 3); this means that in the following RTT_{net} the amount of memory waiting to be traced is again bounded by the same value. In fact, packets do not need to wait to be traced if the RTT_{net} expires (e.g. if an ACK has been lost or a congestion has occurred), this safely means that RTT_i^{min}, for all flows i, will not be updated in this sampling period. Consequently, the amount of memory used by PINK

is $\mathcal{O}(b)$ for the queue and $\mathcal{O}(b)$ for the tracing, thus concluding the proof in a cumulative linear space complexity of $\mathcal{O}(b)$.

Corollary 3. *PINK is computationally efficient, with $\mathcal{O}(b)$ space complexity and $\mathcal{O}(1)$ time complexity.*

Proof. By Corollary 2 we know that PINK is efficient in terms of used memory, which is $\mathcal{O}(b)$. Considering Algorithms 2, 3 and 4 together with Equations 1 and 2, it is easy to infer that the number of instructions required for this executions is constant and no cycles are computed. This way, we can claim that PINK is also efficient in terms of time complexity, thus concluding the proof.

From an experimental point of view, Theorem 1 manifests a stable behaviour even in highly dynamic environments. In fact, when a new flow starts to transmit, a SYN packet is sent through the network and captured by PINK; the algorithm increases the number of active flows and starts to update the RCV.WND of each ACK according to Equation 1, scaling down the bandwidth of each flow in order to accommodate the new one. Conversely, when a flow stops to transmit, a FIN packet is sent through the network and captured by PINK; this time, the algorithm decreases the number of active flows and increases the RCV.WND of each ACK according to Equation 1, increasing the bandwidth of each active flow in order to continue to efficiently exploit the bottleneck link capacity.

The only shortcoming of this description is represented by SYN packets, that are not contemplated by Theorem 1. These packets, in fact, belong to flows that are not currently part of the proof. This highlights the main current weakness of PINK: it is not resilient to drops caused by SYN flooding Denial of Service attacks. If a group of malicious flows would start to flood the network with SYN packets, the analytical bounds provided by Theorem 1 do not hold anymore and some packet drops can be experienced. Anyway, the general high level performance of PINK can be maintained by using a sampling period (as it is the case in other AQM algorithms, e.g. GREEN [9]) in which PINK monitors the TCP flows that effectively transmit data; in other words, only those are

Table 2: Experimental network setup for DSL testbed: the underlined parameters are taken from [15]

Model	ns3-2.24
Access/Remote Network	Ethernet 100 Mbit/s
Bottleneck Network	Ethernet 10 Mbit/s
Base RTT	100 ms
Gateway Queue	125KB
MTU	1500 Bytes
MSS	1000 Bytes
Active Clients	2, 4, 8, 16, 32, 64
AQM	DropTail, ARED, CoDel, GREEN, PINK

Table 3: Experimental network setup for SAT testbed: only changes from Table 2 are reported

Bottleneck Network	Ethernet 4 Mbit/s
Base RTT	700 ms
Gateway Queue	350KB

considered in Equation 1, while the malicious flows that transmit SYN packets only are identified by the PINK computations and subsequently banned. This sampling period could be activated only when a packet drop occurs, in order to avoid introducing any overhead in general scenarios while shielding PINK from SYN flooding attacks. Another possible solution consists in coupling PINK with the algorithm proposed in [24], an AQM techniques that mitigates congestion due to DoS attacks, thus preventing SYN flooding problem.

5. Simulation Environment

In order to demonstrate the effectiveness of PINK, we developed two main testbeds. The first one is compliant to the one used in [15], hereafter referred to

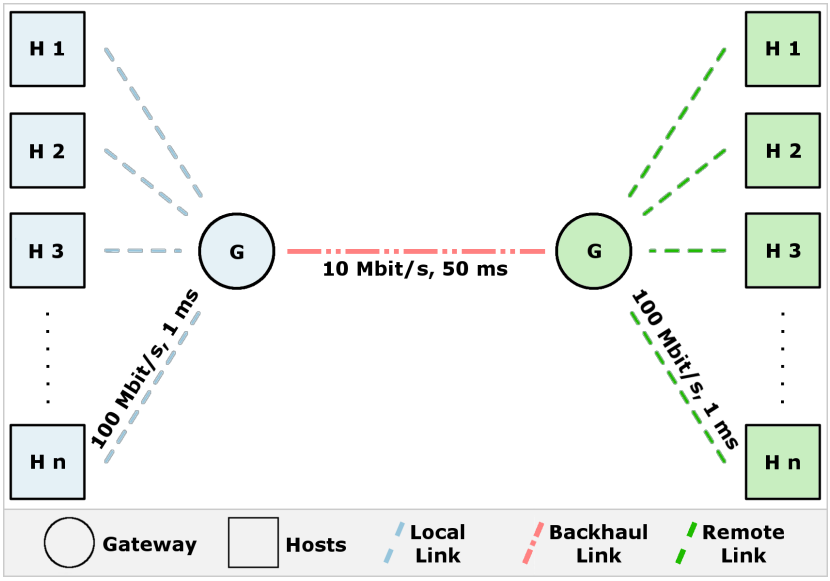

Figure 5: Network DSL testbed topology.

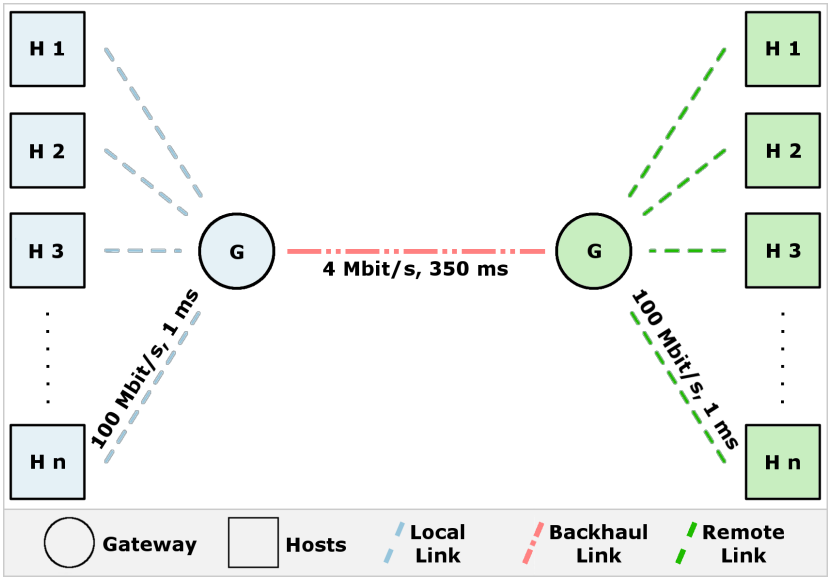

Figure 6: Network SAT testbed topology.

as "DSL testbed", while the second one is an equivalent version of the former designed for Satellite environments, hereafter referred to as "SAT testbed". As simulation platform, we used the ns-3 network simulator; PINK source code, with instructions to reproduce our results, is available at [25]. The parameters used in our simulations for the DSL testbed are summarized in Table 2, while Table 3 reports only the differences that pertain to the SAT testbed; basically, each testbed represents two distinct networks connected with a bottleneck link, accessible to the end hosts of each network through a border gateway, as depicted in Figure 5 for the DSL tesbed and Figure 6 for the SAT tesbed, respectively. The topology of this two testbeds is the same, the only differences belong to the bottleneck link, in particular we are referring to the propagation delay and the bottleneck bandwidth. This choice let us to focus on two of the main parameters that influence the behaviour of an AQM algorithm, the local bandwidth of the link on which it operates, and the global RTT of the connections. By doing this, it is possible to clearly evaluate the impact, on the AQM performance, of this two parameters only without introducing any other environmental condition that could alter the analysis.

We decided to compare PINK with four Queue Manager algorithms, listed as follows together with a brief explanation of the rationale behind each choice.

(i) DropTail, as it can be used as a benchmark to represent the default solution when no AQM technique is in place.

(ii) CoDel, as it can be considered the state of the art.

(iii) ARED, due to the performance provided in [15].

(iv) GREEN, due to its rate-based approach that is shared from PINK.

We ran various experiments consisting in a set of backlogged TCP flows, created between pairs of end hosts in which sender and receiver belong to different networks. Common parameters for the experiments are the bottleneck maximum queue size, set equal to the network Bandwidth Delay Product of the

bottleneck link (125 KB for the DSL case, 350 KB for SAT), and their duration, set to 300 seconds. During this amount of time, the sender applications continuously generate data to be transmitted with TCP protocol. We describe comparisons and outcomes, along with specific experiment parameters, in the next Section.

6. Performance

In this section we analyse the performance of PINK in many different environments, comparing its results with the other well known AQM algorithms described in Section 5. As simulation platform, we used the ns-3 network simulator.

6.1. DSL environment

The discussion starts by providing some numerical results obtained by simulating the DSL testbed, whose simulation parameters are detailed in Table 2. We present the results on goodput, an analysis on per-packet RTT distributions, a flow fairness evaluation and a simple drop analysis.

6.1.1. Goodput

Figure 7 shows the achieved goodput of DropTail, ARED, CoDel, GREEN and PINK for different levels of congestion at the bottleneck link. Higher values denote higher performance. The congestion degree is proportional to the number of the active nodes, which varies from 4 up to 64. All the AQMs under test are configured with their default parameters (e.g. *target_delay* of 5ms for CoDel). By looking at the Figure 7, as first remark we claim that the results are compliant to [15], recording a minor difference on the goodput exploitation of CoDel that is slightly better with respect to ARED. The latter, in fact, suffers a bit, especially in congested scenarios. Considering instead well-known algorithms, CoDel behaves very well, achieving a goodput level comparable to those of DropTail which is the benchmark for this figure of merit. Furthermore, the

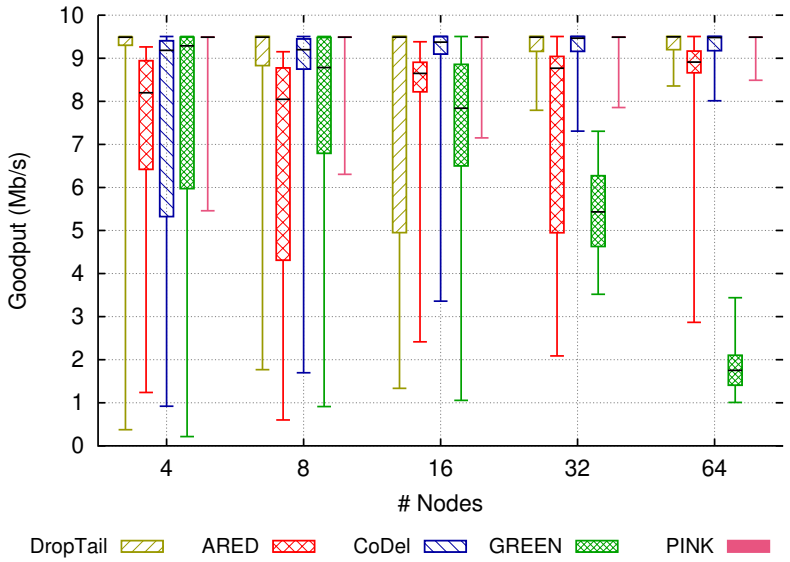

Figure 7: Application goodput at the bottleneck link (per 1-sec intervals). Different numbers of active nodes with RTT_{base} = 100ms; bottom and top of whisker-box plots show the 10th and the 90th percentiles, respectively. The PINK whisker-box is flattened on top of the figure. Higher is better.

10th and the 90th percentiles of CoDel goodput tend to approach each other as the number of nodes increases.

On the contrary, the performance of GREEN are very poor. With the values of bandwidth and RTT_{base} (see Table 2) adopted in the experiments, the drop probability (which is also related to the amount of active nodes[2]) increases too much, resulting in a goodput degradation.

Results even higher than DropTail and CoDel, and considerably better than GREEN, are those achieved by PINK. With the latter, in fact, the goodput is extremely stable with the 10th and the 90th percentile almost identical and close to the optimal value[3], with the bottom of the candlestick that increases as a function of the congestion level.

It is clearly undesirable to have a higher goodput if it comes at the cost of an increased application delay: in the following, we investigate the delay perceived at the application level (through an RTT analysis) to see if PINK increases the RTT of the flows.

6.1.2. RTT analysis

Figure 8 shows the per-packet RTT distributions of AQMs for different congestion levels, where lower values (and lower variances) indicate better performance. As a side note, also in this test our CoDel and ARED results are compliant with [15]. With regards to queuing delay, DropTail represents a negative benchmark of what happens on a bottleneck link when no AQM techniques are employed. The RTT_{base} derived from the sole propagation delay of this channel is 100ms, but by considering in addition the transmission time and the delay inside LAN networks, the smallest RTT value that has been registered during the simulations is equal to 107ms. On the Figure, the difference between the reported value and the smallest value of the candlesticks (107ms) corresponds

[2]See Equation 2 of [9] for further details.

[3]With a MSS of 1000 bytes, the frames in this simulations have the size of 1052 bytes. This upper-bounds the goodput at 9.5Mbit/s, which is lower than the upper bound of [15]. We have been unable to align to their goodput value since their MSS value is not disclosed.

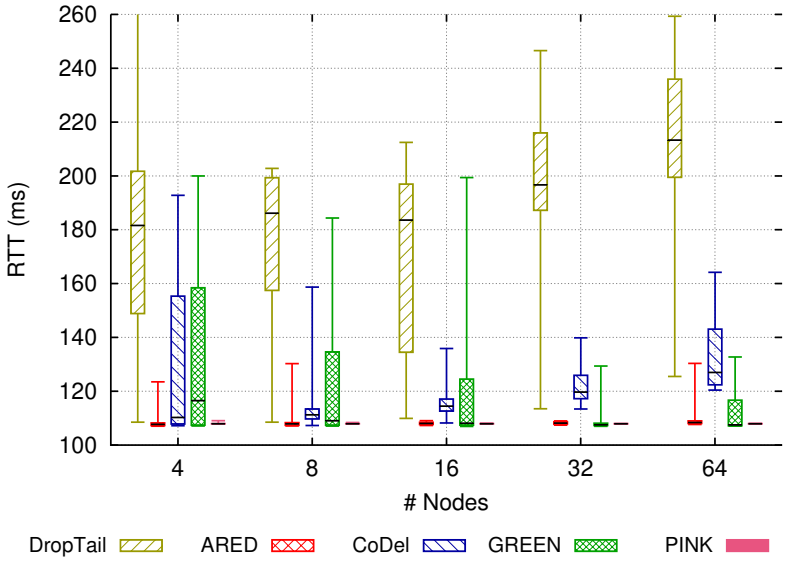

Figure 8: Per-packet RTT. Different numbers of active nodes with $\text{RTT}_{base} = 100$ ms; bottom and top of whisker-box plots shows the 10th and the 90th percentiles, respectively. The PINK whisker-box is flattened on bottom of the figure. Lower is better.

to the queuing delay at the bottleneck.

From this experiment, some observations can be made: CoDel's median, as well as the 10th and the 90th percentiles of queuing delay, increases proportionally to the network congestion level, especially when the number of nodes is above 4. If ARED was suffering the comparison with CoDel from a goodput point of view, here ARED outperforms CoDel, obtaining an almost stable queuing delay; this is especially true for the median, with the values of the 10th and the 90th percentile that are close to the lower bound.

Likewise, GREEN performs better with respect to the queuing delay performance metric. In fact, it provides a trend which is the CoDel opposite; the median, the 10th and the 90th percentiles decrease as a function of network congestion.

The main result of this Figure, however, is the performance of PINK. Firstly, it is extremely stable, with all the candlesticks collapsed and close to the lower bound value. Coupled with a higher goodput, this indicates that PINK is allow-

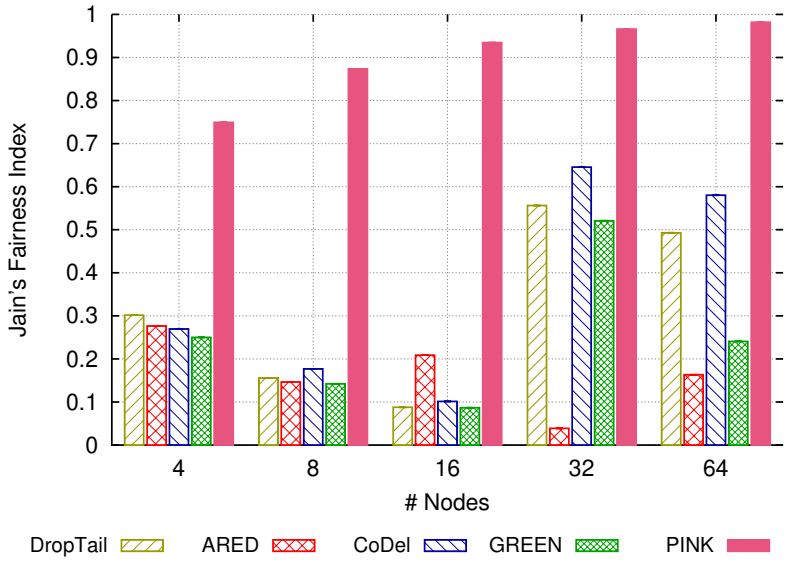

Figure 9: Worst-case Jain's Fairness Index calculated per 1-sec intervals, varying the active nodes number.

ing the exploitation of the bottleneck channel by TCP flows without introducing any unwanted delay.

The next question to investigate is the following: is PINK able to serve equally all flows, or some of them unfairly hold more resources than others? And if that is the case, what is the proportion of the unfair bandwidth distribution?

6.1.3. Fairness

In order to evaluate the fairness between different TCP flows when coupled with different AQM algorithms, we considered the Jain's Fairness Index. Assuming equal data rate requirements among flows, the instantaneous Jain's Fairness index JFi is defined in terms of the instantaneous data rate R_i as:

$$JFi(R_1, R_2, \ldots, R_n) = \frac{(\sum_{i=1}^{n} R_i)^2}{n \cdot \sum_{i=1}^{n} {R_i}^2}$$

where n is the number of active flows, R_i is the instantaneous data rate of flow i and JFi is a real number in the interval $[\frac{1}{n}, 1]$ with a maximum best-case value of 1, if the achieved rate is equal for all flows, and a minimum worst-case value

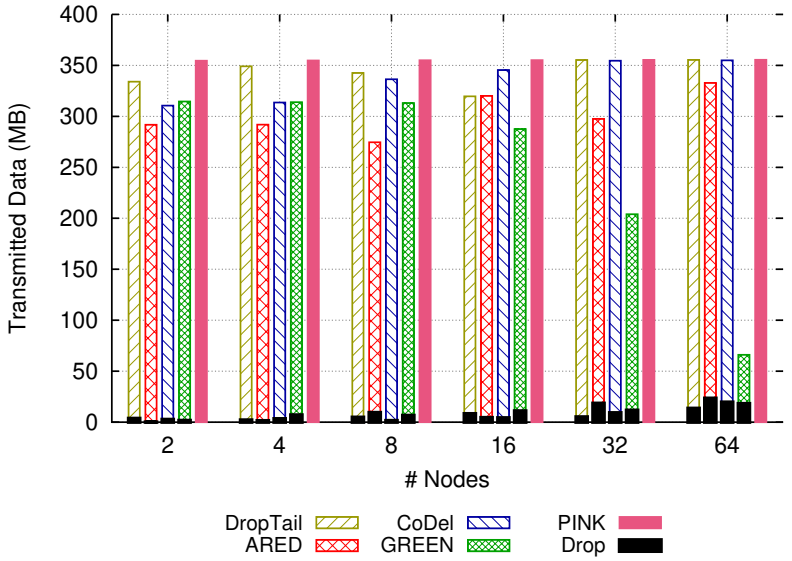

Figure 10: Transmitted data at the end of the experiment on the bottleneck link overlapped by the dropped data.

of $\frac{1}{n}$ if only one aggressive flow is filling the entire data rate of the system.

We reported in Figure 9 (higher values indicates better performance) the worst-case JFi calculated during each simulation, for the different AQM techniques and for different network congestion levels. At a first look, it emerges that ARED is particularly unfair, without exhibiting a clear variation pattern as a function of the congestion level. On the contrary, DropTail, GREEN, and CoDel suffer to maintain fairness with low congestion levels, while they increase the performance when moving towards a more congested environment (e.g. with 32 or 64 active flows). Remarkable results are obtained by PINK: in fact, it presents an excellent scalable behaviour, approaching the best-case fairness value of 1 (which means that all flows have the same rate) as the network congestion level increases.

6.1.4. Drop analysis

Last but not least, we analysed the AQMs behaviour in terms of packet drops. This test has been performed in order to analyse the waste of energy

and processing time for managing packets that, eventually, will be lost. Indeed, dropped packets at the gateway have been transmitted over the LAN (consuming resources) and processed by the gateway itself. Figure 10 shows the amount of successfully transmitted data at the end of experiment over the bottleneck link, paired with the amount of dropped data by the bottleneck AQM. The upper-bound of the successfully transmitted data at the application layer, considering a MSS of 1000 Bytes, a gateway frame of 1052 Bytes, a bottleneck link of 10 Mbit/s and a simulation of 300 seconds is equal to 356 MB as result of $\frac{10 \cdot 10^6}{8} \cdot 300 \cdot \frac{1000}{1052}$.

The ratio between dropped and transmitted data is approximately the same for all the AQMs, except for GREEN that has a lower transmitted data value as a consequence of a lower goodput. The amount of drops grows as a function of network congestion and approaches the 10% of goodput, except for the flows employing PINK that indirectly avoid packet drops (as explained in Section 3). This trend poses PINK in a new position with respect to other AQMs, with an emphasis on efficiency thanks to the fact that it exploits the channel capacity without dropping packets and wasting energy.

6.2. Different RTTs

In this subsection we modify the DSL testbed in order to analyse the performance of the different AQM algorithms in presence of flows with different RTTs. These experiments are easier to describe by starting from the DSL testbed, as the only differences are:

- The number of active clients/flows is 4.
- The RTT_{base} for each flow is not fixed to 100ms anymore but the 4 nodes have respectively 100, 150, 200 and 250ms of RTT_{base}.
- Each node transmits 80MB of TCP data.
- The nodes do not start all at the same time but respectively at 0, 25, 50 and 75 seconds of simulated time.

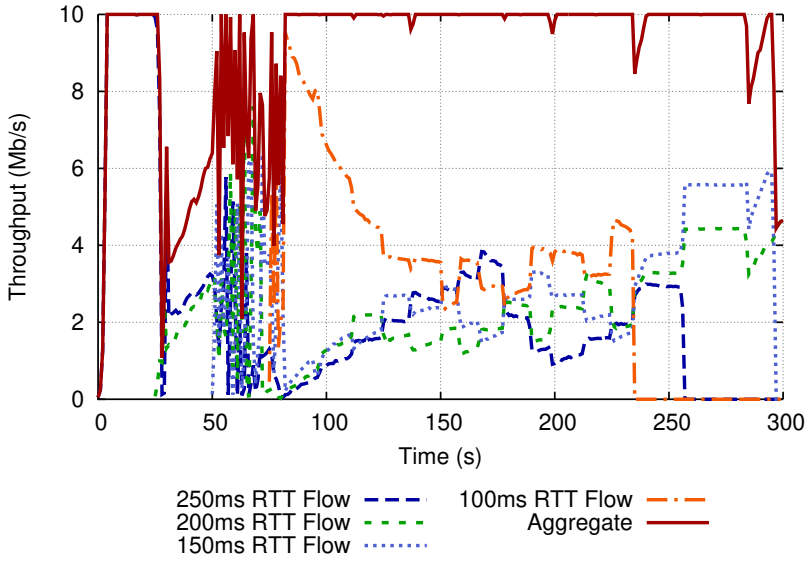

Figure 11: Throughput of 4 different RTT flows at the bottleneck link with DropTail as AQM algorithm.

We decided to modify the DSL testbed by accommodating the aforementioned modifications because it is important to provide a clear picture of what happens on the bottleneck link when flows with different RTTs compete for the transmission. In particular, we aim to provide a clear overview on how the different AQMs behave under these conditions.

We begin the description by commenting the DropTail results in Figure 11. The first 25 seconds of simulations with only one flow active are "safe", while from the 25th to the 75th seconds of simulation the DropTail algorithm is not able to support the flows in maintaining a proper link exploitation, which results in an overall throughput degradation. From the 75th second up to the end of the experiment the aggregate throughput is stable and the link potential is almost completely exploited; the inter-flow fairness, however, is not symmetrical among flows and it is not possible to extract a clear tendency when considering the throughput as a function of the RTT.

Different conclusions can be drawn from Figure 12 where the same experiment is reproduced with CoDel in place. The first 25 seconds of simulation are

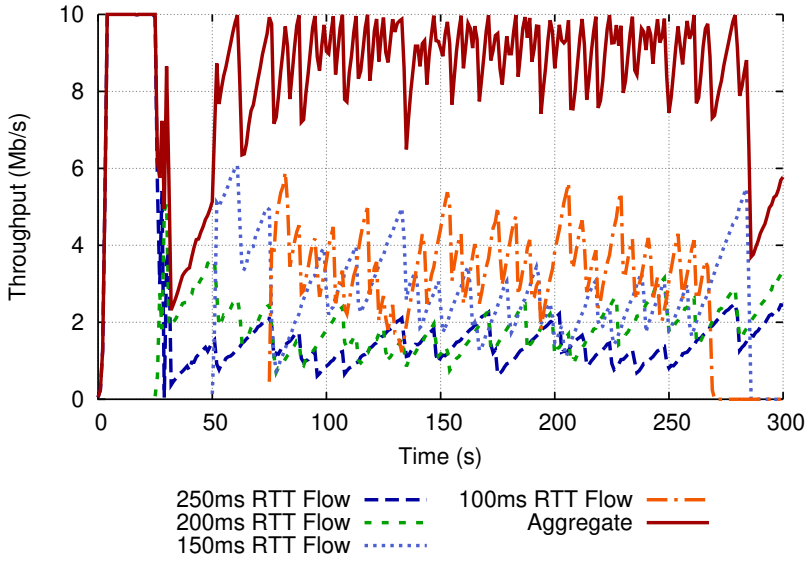

Figure 12: Throughput of 4 different RTT flows at the bottleneck link with CoDel as AQM algorithm.

very similar to the experiment with DropTail, as is the degradation corresponding to the entrance of the flow with a 200ms RTT. From the 50th second up to the end of the simulation (in which a couple of flows successfully complete the 80MB transmission) the aggregate throughput oscillates close to the optimal value. In this large central part of the experiment it is easy to see the general AQM behaviour in terms of RTT fairness; the distribution of bandwidth among the different flows is almost stable, and it is clearly possible to notice how the 100ms-RTT flow obtains more resources if compared to the 250ms-RTT flow. In general, the experiment confirms that CoDel is not able to provide flow fairness when different RTTs are in place, guaranteeing more bandwidth to flows with smaller RTTs. We do not report the ARED results because of their strong similarity to CoDel, as they do not introduce any significant difference.

Figure 13 demonstrates instead the main feature of GREEN. If with the testbed of the previous subsection, in fact, GREEN does not provide good results, this AQM is able to guarantee a stable bandwidth subdivision in presence of flows with different RTTs, as testified by the second half of Figure 13, behav-

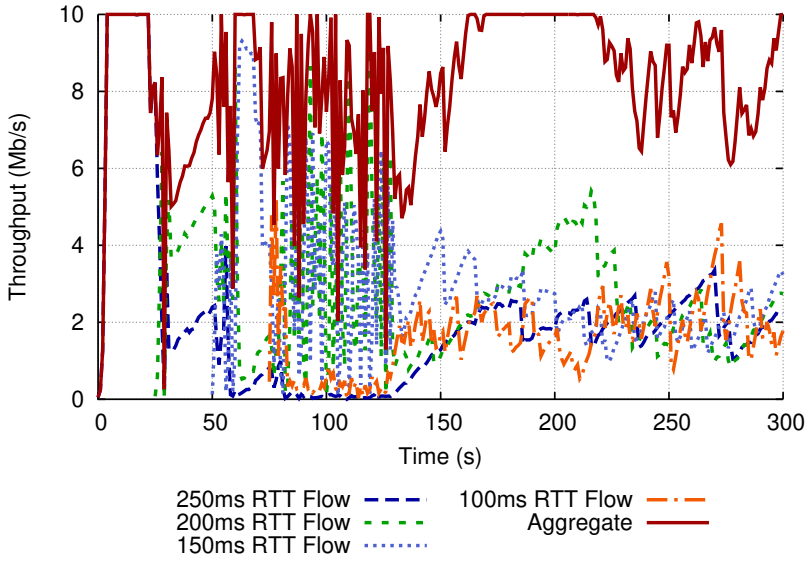

Figure 13: Throughput of 4 different RTT flows at the bottleneck link with GREEN as AQM algorithm.

ing better than CoDel. Unfortunately, the same cannot be claimed for the first part of the experiment, in which GREEN causes strong throughput oscillations even after several seconds from the entrance of the last node.

Finally, Figure 14 reports the performance obtained with PINK. The first thing to note is that the aggregate throughput is almost always firmly close to the optimal value, even when a new node joins the network, a critical moment for all the other AQMs that have been tested. The second thing to note is that all the active flows *always* equally share the bandwidth regardless from each single flow RTT. A third and final thing to note is that all the flows correctly conclude the 80MB transmission before the end of the simulation, a behaviour very close to the optimal and ideal transmission[4]. This experiment underlines, on one side, that the fairness guarantees of PINK are resilient to the RTT variation and, on

[4]Each flow transmits 80MB, i.e. an overall of 320MB is transmitted when considering the four flows; without considering the protocols overheads, at least 260 seconds are required to transmit 320MB on a 10Mbit/s channel.

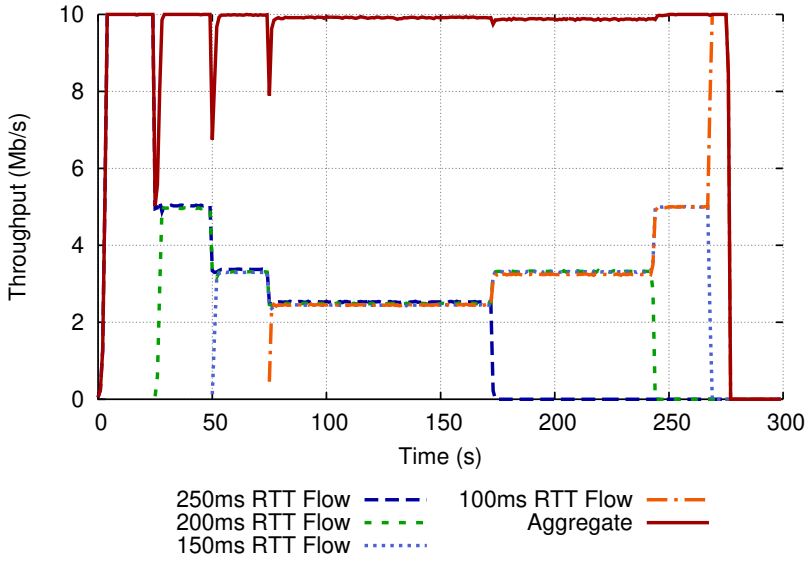

Figure 14: Throughput of 4 different RTT flows at the bottleneck link with PINK as AQM algorithm.

the other, that PINK allows an efficient exploitation of the link resources by permitting to complete the transmissions before the other AQMs, with a time close to the optimal value.

6.3. SAT environment

The following experiments are carried out on a Satellite testbed elaborated from [26], which differs from the DSL testbed for the parameters reported in Table 3, i.e. the base RTT of 700ms and the bottleneck bandwidth of 4Mbit/s. This analysis has been carried out for high bandwidth-delay products pose critical challenges to AQM algorithms, and we aim to emphasize that PINK is able to provide fairly stable high-performance even in this challenging conditions. For this numerical analysis, we also present the results on goodput, an analysis on per-packet RTT distributions, a flow fairness evaluation and a simple drop analysis.

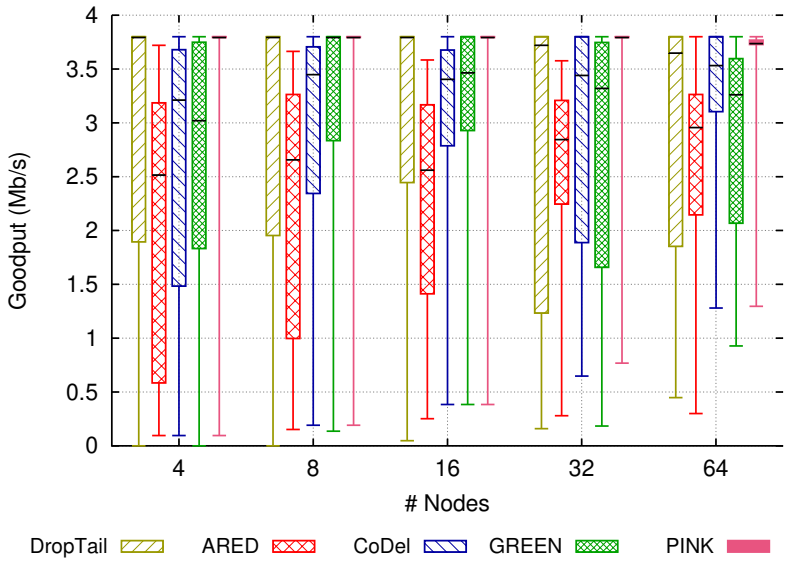

Figure 15: Application goodput at the bottleneck link (per 1-sec intervals). Different numbers of active nodes with RTT_{base} = 700ms; bottom and top of whisker-box plots show the 10th and the 90th percentiles, respectively. The PINK whisker-box is flattened on top of the figure. Higher is better.

6.3.1. Goodput

Figure 15 shows the achieved goodput of DropTail, ARED, CoDel, GREEN and PINK for different levels of congestion at the bottleneck link. Higher values denote higher performance. The congestion degree is proportional to the number of active nodes which varies, as in the DSL testbed, from 4 up to 64. By looking at the Figure, is it possible to notice how the whisker-boxes of almost all the AQMs under exam are wider if compared to Figure 7. This is caused by the more challenging parameters of the SAT testbed with respect to the DSL testbed; in particular, the performance degradation is related to the higher RTT.

Also in this case, the DropTail algorithm could be identified as a benchmark within the group of known algorithms. The high delay only enlarges the whisker-box a little and slightly moves the average goodput under the optimal value of circa 4.8 Mbit/s[5] in a congested environment of 32 or 64 nodes. ARED is probably the algorithm which suffers more the switch from low to high RTTs, as both the box breadth and the average goodput levels are far from the optimal value, aggravating the poor performance achieved by ARED in the DSL testbed in terms of goodput.

The SAT testbed enhances, in a sense, the robustness of CoDel, as it suffers with a minor impact the goodput degradation on a higher RTT network. Although its goodput boxes are slightly wider when compared to the DSL environment, CoDel remains scalable, a property confirmed by the fact that as the congestion level increases the variance is reduced and the average goodput increases. Surprisingly, the GREEN algorithm obtains benefits from the higher RTTs, achieving performance comparable to CoDel. This is due to the drop probability computation which is fairly more stable due to the higher RTTs, avoiding excessive drops and improving the goodput figures.

PINK goodput is extremely stable with the 10th and the 90th percentiles almost identical and close to the optimal value, with the bottom of the candle-

[5]The MSS is again set to 1000 bytes, and consequently the frame is sized 1052 bytes, limiting the optimal throughput to 4.8Mbit/s.

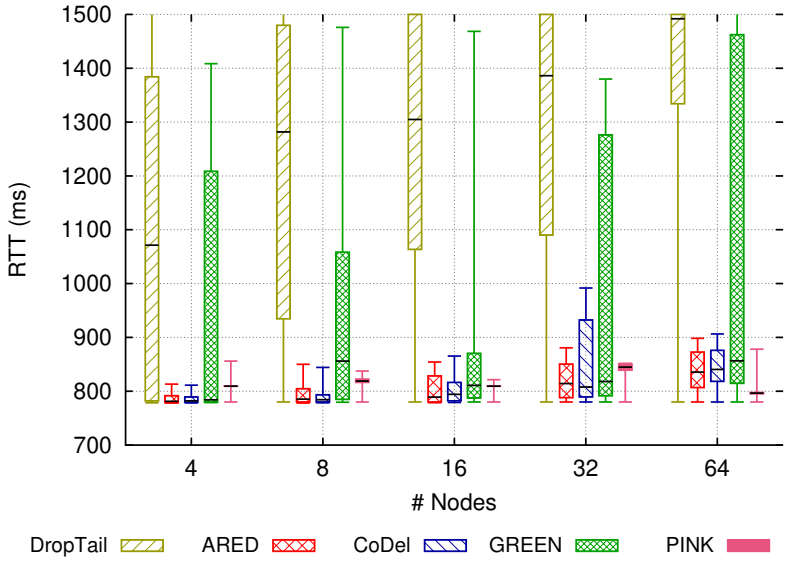

Figure 16: Per-packet RTT. Different numbers of active nodes with RTT_{base} = 700 ms; bottom and top of whisker-box plots shows the 10th and the 90th percentiles, respectively.

stick that increases as a function of the congestion level. This trend highlights the stability of PINK, that maintains remarkable results even in this challenging environments.

We now continue the description of the different algorithms performance through a RTT analysis, investigating also the amount of packets that are stored on the gateway queue.

6.3.2. RTT analysis

Figure 16 shows the per-packet RTT distributions of AQMs for different congestion levels on the SAT testbed, where lower values (and lower variances) indicate better performance.

DropTail represents again a negative benchmark of what happens on a bottleneck link when no AQM techniques are employed. In particular, with respect to the DSL testbed, here we have higher RTTs and larger queue sizes (due to the higher bandwidth-delay product), and the queue overfilling has a huge negative impact on the network performance. In this scenario with a RTT_{base}

of 700ms, we obtain a minimum total RTT value of 780ms when considering the minimum processing and transmission times for a packet. As before, the difference between the reported value and the smallest value of the candlesticks (780ms) of Figure 16 corresponds to the queueing delay at the bottleneck.

In this environment CoDel and ARED behave almost the same way, with good RTT distributions that only tend to increase in absolute value and in variance (with wider boxes) as a function of the network congestion. Only PINK achieves results comparable to ARED and CoDel with the remarkable characteristic to have an almost constant trend as a function of the network congestion and an extremely accurate RTT distribution with very narrow boxes.

Unfortunately, if GREEN seems to behave better on the SAT testbed with respect to DSL in terms of goodput, the same cannot be claimed in terms of RTT variation; GREEN, in fact, provides always higher average RTTs than the other AQMs with a growing trend for both the box size and the median value.

With the next experiment we analyse which fairness guarantees may be provided by PINK and the other AQMs in high RTT networks.

6.3.3. Fairness

The simulations reported in Figure 17 show that the worst-case JFi continues to enhance, in a sense, the remarkable performance of PINK in terms of fairness, while showing the limitations of the other AQMs. The performance of DropTail and GREEN are on the same wave of the figures obtained in the DSL simulations and reported in Figure 9. Some little improvements can be registered by ARED and CoDel, as the former manifests a fairness degradation that is function of the congestion while the latter maintains a stable JFi (apart for the simulation with 32 active nodes with which a poor fairness balance is exhibited). The performance of PINK are almost constant and close to the 0.8 value.

6.3.4. Drop analysis

A consideration about the packet drops is worth to be done also for the high RTT SAT testbed, in which the impact of a drop might be higher. Figure 18

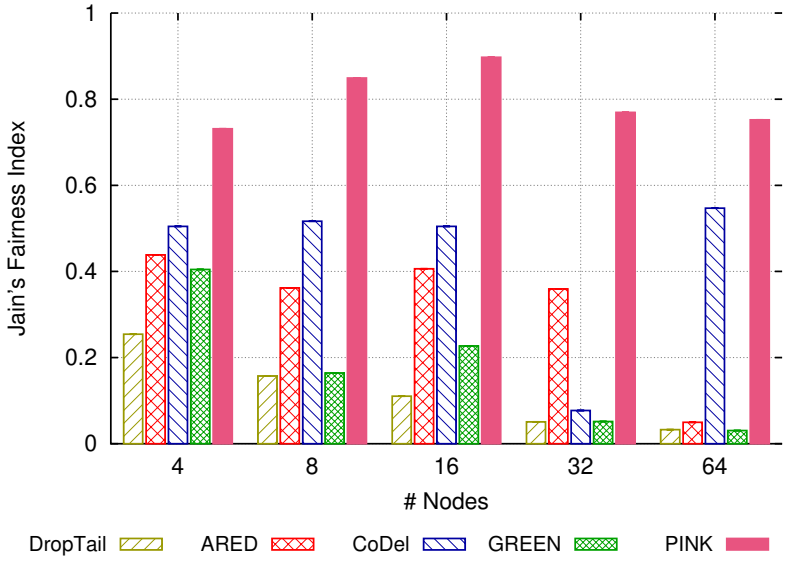

Figure 17: Worst-case Jain's Fairness Index calculated per 1-sec intervals, varying the active nodes number.

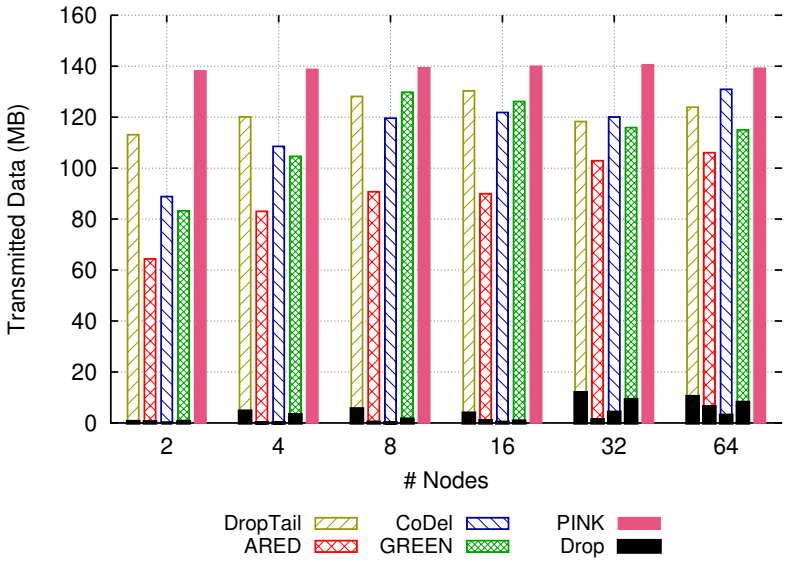

Figure 18: Transmitted data at the end of the experiment on the bottleneck link overlapped by the dropped data.

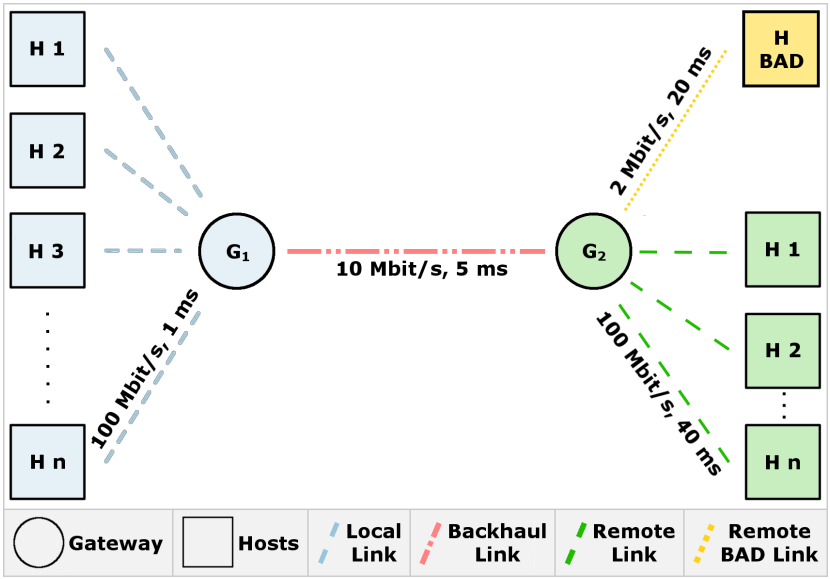

Figure 19: Network DSL testbed topology with multiple bottleneck.

shows the amount of successfully transmitted data over the bottleneck link at the end of the experiment, paired with the amount of dropped data by the bottleneck AQM. The upper-bound of the successfully transmitted data at transport layer, considering a MSS of 1000 Byte, a gateway frame of 1052 Byte, a bottleneck link of 4 Mbit/s and a simulation of 300 seconds is equal to 146 MB as result of $\frac{4 \cdot 10^6}{8} \cdot 300 \cdot \frac{1000}{1052}$.

The amount of drops grows as a function of network congestion and approaches the 10% of goodput, except for the flows that employ PINK that indirectly avoid packet drops (as explained in Section 3). The situation is even more positive for PINK because it not only is the sole AQM with the fewer drops, but it is also the more stable in terms of goodput and the sole able to transmit an amount of data close to the theoretical maximum value of 146MB.

6.4. Multiple bottlenecks analysis

In this Subsection we analyse a scenario that has been built starting from the DSL-environment by adding a multiple bottleneck, this new scenario is depicted in Figure 19. In this testbed, PINK operates on all the interfaces, and one of the

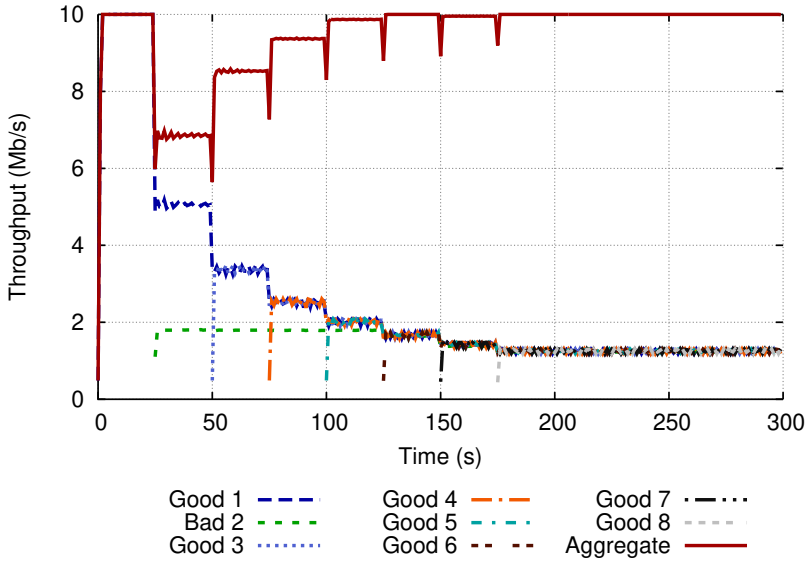

Figure 20: Throughput at the DSL bottleneck link with 8 flows: PINK operates without the resource reassignment algorithm

remote nodes is highlighted as *bad* because of its local bottleneck of 2 Mbit/s. This bottleneck could be physical, e.g. a limited bandwidth service provided by the ISP, or logical due to the PINK assignments on the remote part of the network.

This testbed, coupled with the simulation that follow, is introduced in order to show the resource reassignment algorithm computed by PINK. The simulation is performed with 8 TCP flows, continuously backlogged, between the local and the remote networks. Each flow start each 25 seconds of simulated time. The second flow, that starts at the 25th second, is the only one directed to the *bad* remote link.

In Figure 20 is depicted how PINK behaves if a the resource reassignment algorithm is disabled. When the 2nd flow join the network, the PINK algorithm operating on the remote network assign to its ACK packets a RCV.WND, according to Equation 1, that is calculated in order to let it fill the 2 Mbit/s bottleneck. When the ACK packets arrive on the local Gateway, the blue one on the left of Figure 19, the PINK algorithm that runs on this node has to

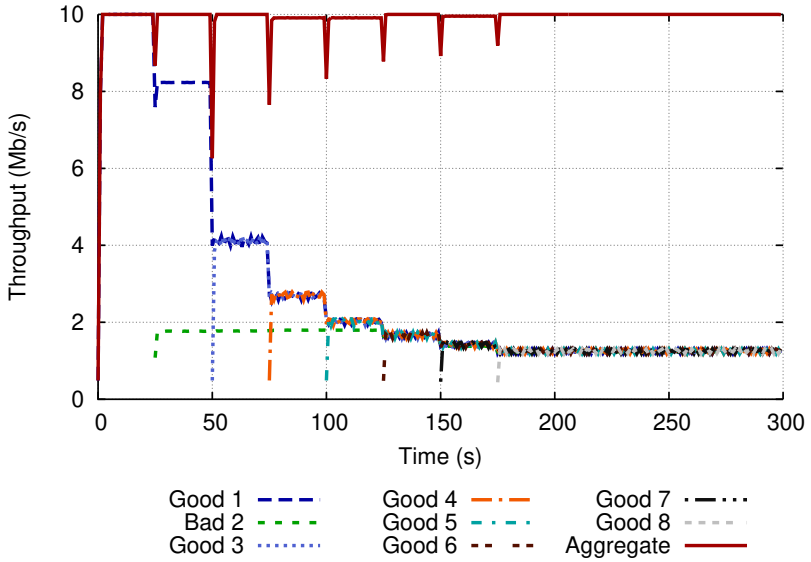

Figure 21: Throughput at the DSL bottleneck link with 8 flows: PINK operates with the resource reassignment algorithm

divide equally the bandwidth with the two active flows, 5Mbit/s each. The first flow is able to deal with it, while the RCV.WND computed for the *bad* flow is bigger than the RCV.WND previously computed by the former PINK operating on the remote bottleneck. That leads to an underutilization of the DSL bottleneck. The situation gets better when at least five flows are active on the local bottleneck link and the *bad* flow REV.WND can be updated with a smaller value, that restore the fairness.

The same simulation is performed also with the resource reassignment algorithm presented in Section 6.4, the result is depicted in Figure 21. In this case, the behaviour of the *bad* flow is the same, due to the presence of the remote bottleneck, while the behaviour of the local PINK changes, giving more bandwidth to the *good* flows in order to maintain a channel exploitation without wasting resources. This mechanism ends when at least five flows join the network and the *bad* flows start to accommodate (update) the RCV.WND according to the local PINK computation.

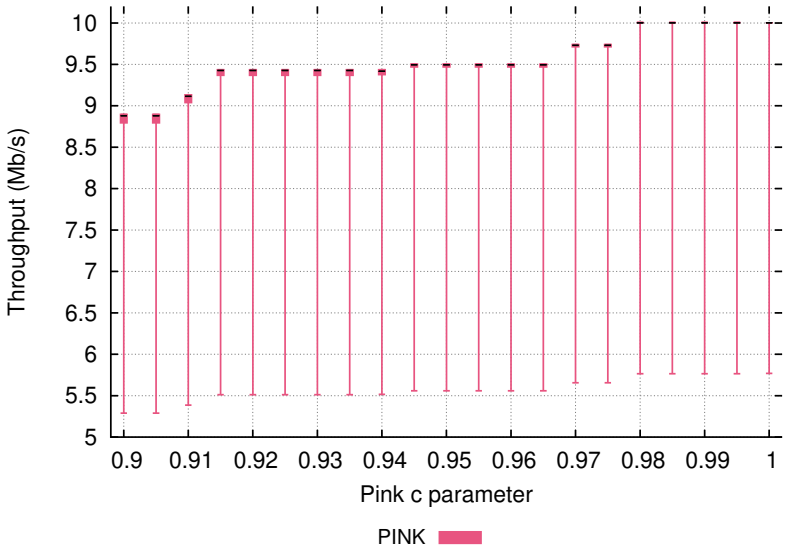

Figure 22: Throughput at the DSL bottleneck link (per 1-sec intervals) with 4 nodes and PINK as a function of c; bottom and top of whisker-box plots show the 10th and the 90th percentiles, respectively.

6.5. PINK constant c analisys

We finally conducted an analysis on the PINK robustness by considering its only tunable parameter, i.e. the exploitation constant c that can have an impact on the performance of the algorithm if rwongly dimensioned. We considered two main figures of merit, one for the DSL testbed and one for the SAT testbed.

One is reported in Figure 22 as the throughput on the bottleneck link of the DSL testbed with 4 active nodes. We chose the throughput figure because the others were almost stable regardless the value of c. Figure 22 shows that the final throughput variance is always well bounded, with narrow candlesticks that confirm the high stability of PINK. At the same time, it is possible to notice how by setting c values that start from 0.92, the throughput already reaches values of 9.5 Mbit/s, almost completely exploiting the channel capacity.

As a second analysis, in Figure 23 we investigated the RTT variation caused by PINK as a function of c in the SAT testbed with 4 active nodes. In a high RTT environment with large buffers is important to contain the RTT variation,

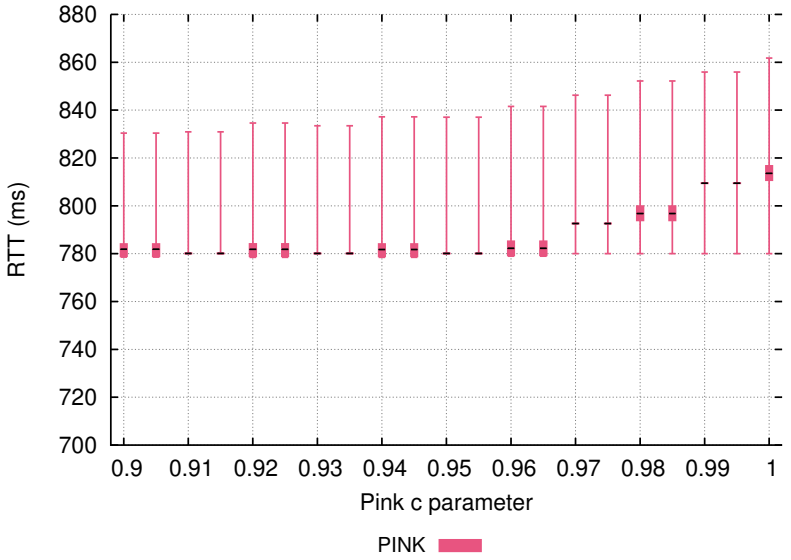

Figure 23: RTT variation at the Satellite bottleneck link (per 1-sec intervals) with 4 nodes and PINK as a function of c; bottom and top of whisker-box plots show the 10th and the 90th percentiles, respectively.

as it can dramatically grow. Figure 23 shows again the narrow candlesticks exhibited by PINK, that reports bounded values with low variance, at the same time showing RTT figures that grow as a function of c; this trend starts only at the end of the range of c with values close to the maximum one, reaching an average throughput of 810ms in the latter.

By considering these last results of PINK performance stability in terms of the exploitation constant c, we can claim that PINK is a very robust algorithm in which the configuration (i.e. the c instantiation) would hardly pose significant issues. Values of c between 0.92 and 0.97 guarantee a good trade-off between throughput exploitation and introduced RTT variation (queue utilization) while is always possible to perform a fine-grained tuning in environments where one figure of merit is considered more important than the other.

7. Conclusions

In this paper we proposed PINK, a queue management algorithm designed for access networks that transparently prevents congestion by imposing a maximum upper bound on the data rates of each client. This is done through the update of the related RCV.WND values in the acknowledgement segments, allowing PINK to supervise bandwidth utilization in order to efficiently exploit a bottleneck channel capacity, enforcing fairness among different flows regardless their RTTs, and at the same time maintaining a low queue occupancy on the gateway. Furthermore, these results are achieved without discarding a single packet. Another important aspect to remark is the algorithm scalability, as all the figures of merit that have been analysed reveal that PINK manifests a positive, or at least a constant, trend as a function of the network congestion level. PINK has been introduced and analysed by both a mathematical formulation and a numerical simulation, and it has been compared with the most valuable and promising AQMs available in literature through the simulation of a wide set of possible scenarios.

References

References

[1] Q. Xu, F. Li, J. Sun, M. Zukerman, A new tcp/aqm system analysis, Journal of Network and Computer Applications 57 (2015) 43–60.

[2] H.-J. Byun, J.-T. Lim, Fair tcp congestion control in heterogeneous networks with explicit congestion notification, Communications, IEE Proceedings- 152 (2005) 13–21.

[3] M. Gerla, R. L. Cigno, S. Mascolo, W. Weng, Generalized window advertising for tcp congestion control, European Transactions on Telecommunications 13 (2002) 549–562.

[4] L. Kalampoukas, A. Varma, K. Ramakrishnan, Explicit window adaptation: a method to enhance tcp performance, Networking, IEEE/ACM Transactions on 10 (2002) 338–350.

[5] H. Byun, An explicit window adaptation algorithm over tcp networks using supervisory control, J. High Speed Netw. 17 (2010) 207–218.

[6] J. Wang, L. Rong, Y. Liu, Design of a stabilizing aqm controller for large-delay networks based on internal model control, Computer Communications 31 (2008) 1911–1918.

[7] Z. Rosberg, J. Matthews, M. Zukerman, A network rate management protocol with tcp congestion control and fairness for all, Computer Networks 54 (2010) 1358–1374.

[8] J. Sun, S. Chan, M. Zukerman, Iapi: An intelligent adaptive pi active queue management scheme, Computer Communications 35 (2012) 2281–2293.

[9] W. chun Feng, A. Kapadia, S. Thulasidasan, Green: proactive queue management over a best-effort network, in: Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE, volume 2, 2002, pp. 1774–1778 vol.2. doi:10.1109/GLOCOM.2002.1188503.

[10] J. Li, M. Yuksel, S. Kalyanaraman, Explicit rate multicast congestion control, Computer Networks 50 (2006) 2614 – 2640.

[11] J. Sun, M. Zukerman, Raq: A robust active queue management scheme based on rate and queue length, Computer Communications 30 (2007) 1731–1741.

[12] H.-Y. Wei, S.-C. Tsao, Y.-D. Lin, Assessing and improving tcp rate shaping over edge gateways, IEEE Transactions on Computers 53 (2004) 259–275.

[13] C. Grazia, M. Casoni, M. Klapez, N. Patriciello, Pink: Proactive injection into ack, a queue manager to impose fair resource allocation among tcp

flows, in: Wireless and Mobile Computing, Networking and Communications (WiMob), 2014 IEEE 10th International Conference on, 2015, pp. 132–137.

[14] Y. Gong, D. Rossi, C. Testa, S. Valenti, M. Taht, Fighting the bufferbloat: On the coexistence of aqm and low priority congestion control, in: Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on, 2013, pp. 411–416. doi:10.1109/INFCOMW.2013.6562885.

[15] N. Khademi, D. Ros, M. Welzl, The new aqm kids on the block: An experimental evaluation of codel and pie, in: Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on, 2014, pp. 85–90. doi:10.1109/INFCOMW.2014.6849173.

[16] T. Høiland-Jørgensen, P. Hurtig, A. Brunstrom, The good, the bad and the wifi: Modern AQMs in a residential setting, Computer Networks 89 (2015) 90 – 106.

[17] S. D. Strowes, Passively Measuring TCP Round-trip Times, Queue 11 (2013) 50:50–50:61.

[18] P. Marchetta, A. Botta, E. Katz-Bassett, A. Pescapé, Dissecting Round Trip Time on the Slow Path with a Single Packet, in: Proceedings of the 15th International Conference on Passive and Active Measurement - Volume 8362, PAM 2014, Springer-Verlag New York, Inc., New York, NY, USA, 2014, pp. 88–97. doi:10.1007/978-3-319-04918-2_9.

[19] B. Veal, K. Li, D. Lowenthal, New Methods for Passive Estimation of TCP Round-trip Times, in: Proceedings of the 6th International Conference on Passive and Active Network Measurement, PAM'05, Springer-Verlag, Berlin, Heidelberg, 2005, pp. 121–134. doi:10.1007/978-3-540-31966-5_10.

[20] W. Jiang, F. Ren, Y. Wu, C. Lin, I. Stojmenovic, Analysis of backward

congestion notification with delay for enhanced ethernet networks, IEEE Transactions on Computers 63 (2014) 2674–2684.

[21] F. Yamaguchi, H. Nishi, Hardware-based hash functions for network applications, in: Networks (ICON), 2013 19th IEEE International Conference on, 2013, pp. 1–6. doi:10.1109/ICON.2013.6781990.

[22] M. Casoni, C. A. Grazia, M. Klapez, N. Patriciello, A congestion control middleware layer with dynamic bandwidth management for satellite communications, 2015. URL: http://dx.doi.org/10.1002/sat.1129. doi:10.1002/sat.1129.

[23] J. Gettys, K. Nichols, Bufferbloat: Dark buffers in the internet, Queue 9 (2011) 40.

[24] H. Bedi, S. Roy, S. Shiva, Mitigating congestion based dos attacks with an enhanced aqm technique, Computer Communications 56 (2015) 60–73.

[25] Pink ns-3 source code, http://netlab.ing.unimo.it/sw/PINK.zip, 2015.

[26] C. Grazia, M. Klapez, N. Patriciello, M. Casoni, A. Amditis, E. Sdongos, H. Gierszal, D. Kanakidis, C. Katsigiannis, K. Romanowski, P. Simplício, A. Oliveira, S. Sonander, J. Jackson, Integration between terrestrial and satellite networks: The ppdr-tc vision, 2014, pp. 77–84. doi:10.1109/WiMOB.2014.6962153.