Deadline-aware TCP Congestion Control for Video Streaming Services

Maxim Claeys*‡, Niels Bouten*‡, Danny De Vleeschauwer†, Koen De Schepper†,
Werner Van Leekwijck†, Steven Latré‡ and Filip De Turck*
*Department of Information Technology, Ghent University - iMinds, Belgium
Email: maxim.claeys@intec.ugent.be
†Nokia Bell Labs, Antwerp, Belgium
‡Department of Mathematics and Computer Science, University of Antwerp - iMinds, Belgium

***Abstract*—Video streaming services have continuously gained popularity over the last decades, accounting for about 70% of all consumer Internet traffic in 2016. All of these video streaming sessions have strict delivery deadlines in order to avoid playout interruptions, detrimentally impacting the Quality of Experience (QoE). However, the vast majority of this traffic uses TCP at the transport layer, which is known to be far from minimizing the number of deadline-missing streams. By introducing deadline-awareness at the transport layer, video delivery can be optimized by prioritizing specific flows. This paper proposes a deadline-aware congestion control mechanism, based on a parametrization of the traditional TCP New Reno congestion control strategy. By taking into account the available deadline information, the modulation of the congestion window is dynamically adapted to steer the aggressiveness of a considered stream. The proposed approach has been thoroughly evaluated in both a video-on-demand (VoD)-only scenario and a scenario where VoD streams co-exist with live streaming sessions and non-deadline-aware traffic. It was shown that in a video streaming scenario the minimal bottleneck bandwidth can be reduced by 16% on average when using deadline-aware congestion control. In co-existence with other TCP traffic, a bottleneck reduction of 11% could be achieved.**

I. Introduction

Over the past decades, multimedia services have gained a lot of popularity. This growth is largely due to video streaming services, accounting for about 70% of all consumer Internet traffic in 2016 [1]. For delivery of video streaming services, HTTP adaptive streaming (HAS) has become the de facto standard. These HTTP-based techniques come with some important advantages. Not only is the video content delivered reliably over Transmission Control Protocol (TCP), HAS also allows seamless interaction through firewalls. On the downside however, as the delivery is based on best-effort Internet, HTTP-based techniques are prone to network congestion and large bandwidth fluctuations due to cross traffic. These influences can be detrimental for the Quality of Experience (QoE).

TCP streaming session use a congestion-control strategy to avoid congestive collapse. For this purpose, the total number of unacknowledged packets that may be in transit is limited by means of a congestion window at the sender side. It was shown that using this congestion-control strategy, multiple flows with similar round-trip times (RTTs) eventually converge to using equal amounts of a contended link [2]. However, different types of services can have different requirements. Considering a video streaming service, hard deadlines are associated with each packet in the video stream once the playout is started. When these deadlines are violated, the playout is temporarily interrupted, negatively impacting the QoE. However, fair bandwidth sharing as introduced by TCP is known to be far from minimizing the number of deadline-missing streams [3].

In general, the client applications are aware of the deadlines associated with the requested content. By introducing these deadlines in the network, multimedia delivery can be optimized by prioritizing specific flows. While DiffServ [4] techniques have been proposed for this purpose in the past, this paper focuses on a best effort networking scenario by proposing a deadline-aware congestion control strategy for video streaming services, based on the traditional TCP New Reno congestion control mechanism. The proposed strategy only requires changes at the sender side and reduces to TCP New Reno congestion control in absence of deadline information. Furthermore, for deadline-missing streams, the proposed approach falls back to TCP New Reno congestion control mechanism to avoid congestive collapse. In the proposed approach, deadline information is passed to the transport layer in the request to send data, as is commonly assumed in related work [5]–[7]. This information provides details about (i) the deadline of the current packet, (ii) the final deadline of the stream and (iii) the bit rate of the streamed content. Based on the deadline information and the current throughput measurement, the algorithm dynamically changes the behavior of the congestion window adaptations. In this way, urgent flows can occasionally achieve a higher throughput than their fair share, while other, less urgent flows back off to allow other sessions to increase their throughput.

The contributions of this paper are threefold. First, a parametrization of the congestion avoidance phase of the TCP New Reno congestion control mechanism is proposed. Furthermore, the feasibility of prioritizing streams by changing the configuration of these parameters is demonstrated. Second, a deadline-aware congestion control strategy is introduced, dynamically adapting the parameter configuration of the congestion avoidance phase based on the deadlines of streaming sessions. Third, the performance of the proposed approach

978-3-901882-85-2 © 2016 IFIP

is thoroughly evaluated, both using conceptual examples and large scale packet-based simulations in NS-3. For this purpose, both general traffic configurations and video-on-demand (VoD)-only scenarios are considered.

The remainder of this paper is organized as follows. First, Section II gives an overview of related work on transport protocol optimizations for multimedia delivery. Next, the feasibility of the proposed approach is demonstrated in Section III. The proposed algorithm is introduced in Section IV, while its performance is evaluated in Section V. Finally, Section VI presents some final conclusions.

II. Related work

Multiple transport protocols that offer novel delivery models to improve the support for multimedia applications have been presented in literature, including Stream Control Transmission Protocol (SCTP) [8], Datagram Congestion Control Protocol (DCCP) [9] and Shared Content Addressing Protocol (SCAP) [10]. However, ossification of the transport layer limits the deployability of new transport protocols. Furthermore, these approaches require changes both at the sending and receiving side, as well as in the network.

TCP-RTM proposes extensions to TCP that improve performance of multimedia applications by allowing minimal amount of packet re-ordering and loss in the TCP stack [11]. This approach modifies the interaction between the application and the receiver buffer, rather than proposing modifications to TCP itself. Selective negative acknowledgments are used to allow senders to be informed of segments that were skipped by the application, preventing retransmission. Similarly, TCP Hollywood is a protocol offering unordered, partially reliable message-oriented transport service that is well suited for multimedia applications [12]. While this approach is focused on deployability, the inconsistent retransmission mechanism is visible to middleboxes performing deep packet inspection, which might disrupt these connections. Furthermore, TCP Hollywood requires changes at both the sender and receiver side. While the above approach focus on flow-based optimization, media-TCP-friendly congestion control (MTCC) takes into account the deadlines for each packet [13]. Furthermore, additional complexity is added to this solution by introducing a packet-based multimedia model, considering distortion impacts as well as inter-dependencies between multiple packets, using directed acyclic graphs.

In the area of data center services, multiple transport protocols have been proposed with the main objective of minimizing deadline misses. D^3 introduced the idea of introducing deadline awareness into data center networks by proactively allocating bandwidth before data transmission [5]. PDQ further improves flow completion times compared to D^3 [14]. However, both protocols are incompatible with TCP. Deadline-aware data center TCP (D^{2}TCP) is a TCP-friendly protocol implementing deadline-aware delivery [6]. Similar to our work, deadline-aware congestion avoidance is implemented by changing the adaptation of the congestion window. In D^{2}TCP, the window size is modulated based on the deadlines and the extent of congestion in the network. However, as the approach heavily relies on Explicit Congestion Notification (ECN) feedback and is specifically aimed at data center topologies and services, the applicability for video streaming over public Internet is limited. DSTCP builds on the ideas introduced in D^{2}TCP, adjusting the congestion window size of a flow based on its deadline, its size and the degree of network congestion [7]. As was the case with D^{2}TCP, DSTCP heavily relies on ECN feedback. L^{2}DCT has been presented as a TCP-friendly protocol, reducing completion times of short flows in data center networks [15]. As opposed to these works, the focus in this paper is on multimedia delivery in public Internet, rather than data center networks.

III. Feasibility study

As a starting point for the proposed approach, TCP New Reno [16], an improved version of the traditional TCP Reno congestion control mechanism, is used. For a detailed description of the TCP New Reno congestion control mechanism, the reader is referred to literature. In this work, we specifically focus on the congestion avoidance phase of TCP New Reno while the slow start phase, the fast retransmit behavior and the reaction to timeouts are unchanged. This section proposes a parametrization of the congestion avoidance phase and demonstrates the feasibility of steering the aggressiveness of a TCP stream using this parametrized scheme.

A. Parametrized congestion avoidance

In the congestion avoidance phase, TCP (New) Reno follows an additive increase/multiplicative decrease (AIMD) scheme to adapt the congestion window size. With AIMD, a linear growth of the congestion window is combined with an multiplicative reduction when a congestion event takes place, resulting in the well known TCP sawtooth behavior. More concretely, with TCP (New) Reno, the congestion window $cwnd$ is increased as show in equation (1) for every incoming non-duplicate acknowledgment, where MSS represents the maximum segment size [17]. This adjustment provides an acceptable approximation to the underlying principle of increasing the congestion window with one full-sized segment every RTT. When congestion is detected in the form of triple duplicate acknowledgments, the congestion window is set to $cwnd = 0.5 * cwnd + 3 * MSS$.

$$cwnd = cwnd + \frac{MSS * MSS}{cwnd} \tag{1}$$

It was shown that multiple flows with similar RTTs using the same AIMD congestion control, eventually converge to use equal amounts of a contended link [2]. However, this only holds when these flows use the same AIMD scheme. In this work, we propose to parametrize the AIMD scheme, resulting in an increase of the congestion window as expressed in equation (2) for every incoming non-duplicate acknowledgment. Using this equation approximates the increase of the congestion window with one full-sized segment every $\alpha \in \mathbb{R}^+$ RTTs. Upon reception of the third duplicate acknowledgment, the congestion window size is reduced as shown in equation (3), for $\beta \in]0;1[$. For $\alpha = 1.0$ and $\beta = 0.5$, this AIMD scheme corresponds to the scheme used in TCP (New) Reno.

$$cwnd = cwnd + \frac{MSS * MSS}{\alpha * cwnd} \tag{2}$$

$$cwnd = (1 - \beta) * cwnd + 3 * MSS \tag{3}$$

B. Parameter influence

Intuitively, lower values for α or β, causing a faster increase or slower decrease of the congestion window respectively, result in more aggressive behavior and corresponding higher throughput. However, to study the influence of these parameters in detail, simulations have been performed to compare the throughput of a client using TCP New Reno congestion control to a client using different parameter values.

For this purpose, two scenarios have been considered where respectively two and ten clients simultaneously try to send data at a rate of 10Mbps over a bottleneck link with a capacity equal to half of the sum of requested rates (i.e. 10Mbps and 50Mbps in the two and ten client scenario respectively). To estimate the influence of α, for one of the clients the value of α is varied between 0.05 and 5.0 with a fixed value of $\beta = 0.5$, while the other clients use TCP New Reno congestion control (i.e. $\alpha = 1.0, \beta = 0.5$). Similarly, the influence of β is evaluated by varying its value between 0.01 and 0.99 while keeping a fixed value of α=1.0. Due to the probabilistic nature of the applied random early detection (RED) queuing discipline [18], all experiments have been performed for five iterations, presenting the average results.

Figure 1 shows the influence of the value of α on the ratio between of the throughput achieved by the client using varied parameters and the average throughput achieved by the other clients, using TCP New Reno congestion control. As expected, both in the two and ten client scenario, for low values of α (< 1.0) the achieved throughput is significantly higher than the fair share throughput while the opposite is true for high values of α (> 1.0). It can be seen that for very small values of α (< 0.10), the client becomes too aggressive, resulting in decreased performance. By increasing the congestion window too fast, the client perceives a lot of timeouts, resulting to stream to execute in the slow start phase very often. Furthermore, the throughput ratio converges and the influence is limited when increasing α above 4.0.

Furthermore, Figure 2 shows the influence of the value of α on the total throughput on the contended link when all clients use this specific value, relative to the throughput achieved with TCP New Reno congestion control. It can again be seen that for very small values of α (< 0.1), the clients become too aggressive, causing a fallback to the slow start phase too often. As a result, the flows are not able to fill the link. For values above $\alpha >= 0.5$, the general performance is within 1% of the TCP New Reno performance.

Similarly, the influence of the value of β is presented in Figure 3 and Figure 4. It can again be seen that, as expected, a flow is able to achieve a significantly higher or lower throughput than its fair share by respectively using a value of $\beta < 0.5$ or $\beta > 0.5$. As was the case with α, a general performance degradation can be perceived when all streams become too aggressive (i.e. $\beta < 0.1$). Furthermore, a performance drop of about 10% can be noticed when all streams use high values of β. As this situation only occurs when all streams voluntarily back off, indicating that the available bandwidth is higher than required for all flows, it is unlikely to occur in practice.

Based on the above analysis, it can be summarized that the aggressiveness of a stream can effectively be influenced by varying the value of α between 0.1 and 4.0 or varying the value of β between 0.1 and 0.95.

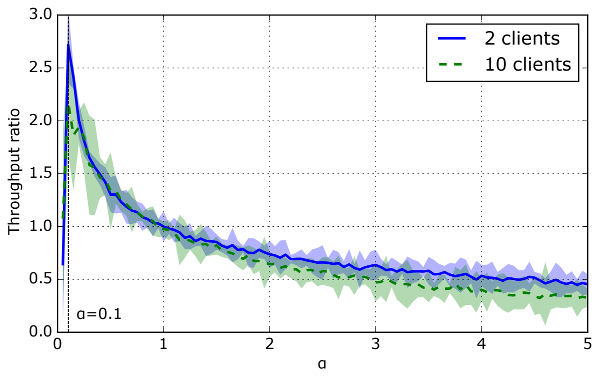


Figure 1. Influence of the AIMD parameter α on the average achieved relative throughput on the contended link. The areas represent the 95% confidence interval.

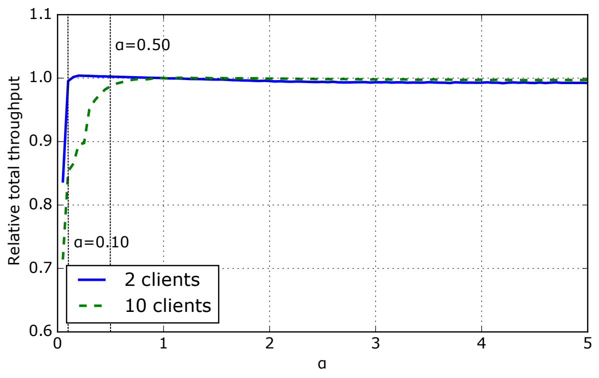


Figure 2. Influence of the AIMD parameter α on the total throughput achieved on the contended link, relative to the throughput achieved with TCP New Reno congestion control.

IV. ALGORITHM

As shown in Section III, the throughput can be increased or decreased by using different values for α or β. In this section, an algorithm is proposed to dynamically adapt the parameter values based on the deadlines of a streaming session. For this purpose, the concept of *deadline margin* is introduced. Consider a video stream s with a bit rate r_s and a total length of l_s seconds. When this stream has a begin deadline d_0, for every byte $b \in [0; \frac{r_s * l_s}{8}[$ the corresponding deadline d_b can be

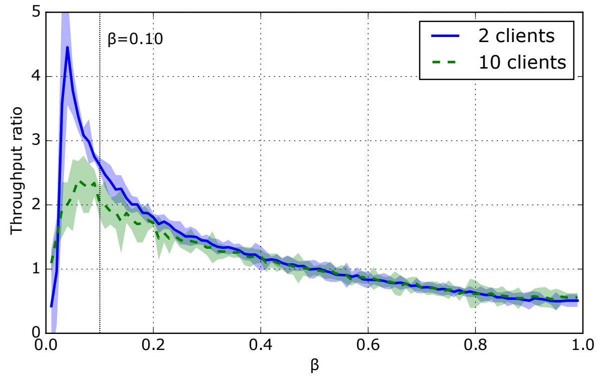


Figure 3. Influence of the AIMD parameter β on the average achieved relative throughput on the contended link. The areas represent the 95% confidence interval.

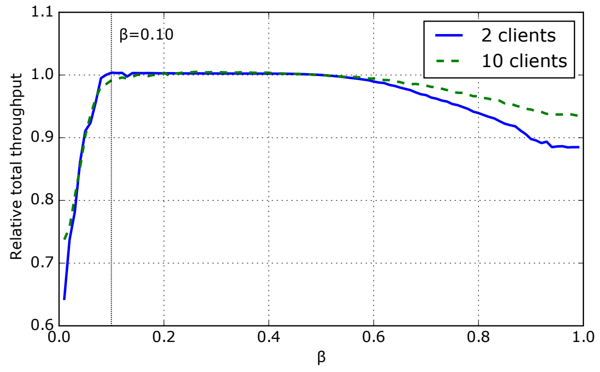


Figure 4. Influence of the AIMD parameter β on the total throughput achieved on the contended link, relative to the throughput achieved with TCP New Reno congestion control.

calculated as shown in equation (4). The deadline of the last byte of the stream is called the end deadline of the stream, denoted as $d_e = d_0 + l_s$.

$$d_b = d_0 + \frac{b * 8}{r_s} \tag{4}$$

When at time t the next byte to be sent is b, the current deadline margin is denoted as $m = d_b - t$. The general goal of the proposed approach is to keep this deadline margin between a lower and upper bound, denoted as m_l and m_u respectively. Both m_l and m_u are parameters of the algorithm which are configured a priori. In order to keep the deadline margin within the predefined bounds, the value of α or β can be changed within their respective ranges $[\alpha_l; \alpha_u]$ and $[\beta_l; \beta_u]$, with a granularity of α_g and β_g respectively. As discussed in Section III, it is suggested to vary α between 0.1 and 4.0 at a granularity of 0.1, or to vary β between 0.1 and 0.95 at a granularity of 0.05. To avoid too much fluctuation in the parameter values and to reduce the computational overhead, it is suggested to only perform parameter updates at fixed time intervals, indicated by the reactivity time t_r. Based on preliminary evaluations, a reactivity time of t_r=1s will be used in the remainder of this paper.

Algorithm 1 Outline of the proposed algorithm to dynamically adapt the parameter value α based on the current deadline margin. This update is performed every t_r seconds.

```
Input:
    $t$: current time
    $T$: current throughput measurement
    $d_b$: deadline of next byte to send
    $d_e$: end deadline of the stream
    $r_s$: bit rate of the stream
 1: $m = d_b - t$
 2: if ($m < m_l$) then
 3:     $\alpha = \alpha_l$
 4: else if ($m > m_u$) then
 5:     $\alpha = \alpha_u$
 6: else
 7:     $rt = d_e - t$
 8:     $rb = (d_e - d_b) * r_s$
 9:     $T_u = \frac{rb}{rt - m_u}$
10:     $T_l = \frac{rb}{rt - m_l}$
11:     if ($T > T_u$) then
12:         $m_\delta = m_u - m$
13:         $\alpha_\Delta = \alpha_l - \alpha$
14:     else if ($T < T_l$) then
15:         $m_\delta = m - m_l$
16:         $\alpha_\Delta = \alpha_u - \alpha$
17:     else
18:         $m_\delta = \infty$
19:         $\alpha_\Delta = 0$
20:     end if
21:     $C = \frac{m_\delta * r_s}{|r_s - T|}$
22:     $\alpha_\delta = \text{round}(\frac{\alpha_\Delta * t_r}{C}; \alpha_g)$
23:     $\alpha = \alpha + \alpha_\delta$
24: end if
```

Algorithm 1 presents the pseudo-code of the proposed algorithm to dynamically adapt the parameter value α. In an identical way, the algorithm can be applied to dynamically change the value of β. This algorithm is executed every t_r seconds, as long as no deadlines were missed for the considered stream. To avoid congestive collapse and to maintain a degree of fairness with other TCP flows, a streaming session falls back to TCP New Reno congestion control (i.e. α=1.0, β=0.5) when a deadline was missed. Based on the current time t and the deadline d_b for the next byte to send b, the current margin m is calculated (line 1). When the current margin is out of the predefined bounds, the value of α is directly updated to the highest (α_u) or lowest (α_l) value to drastically decrease or increase the aggressiveness of the stream respectively (lines 2-5). When the current margin is between the bounds, the upper and lower threshold of the required throughput, T_u and T_l, are calculated (lines 7-10). T_u and T_l respectively denote the highest and lowest required throughput that allows to finish the remainder of the stream without leaving the deadline margin bounds. Their calculation is based on the remaining time rt

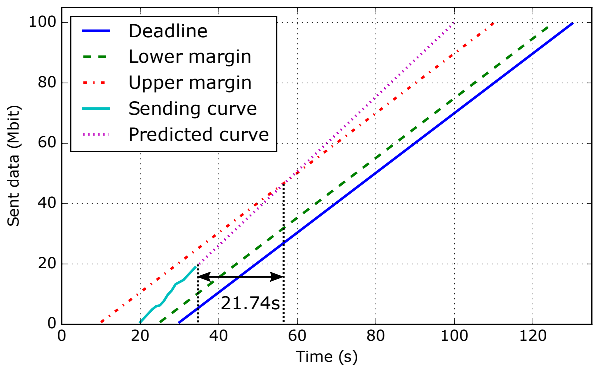


Figure 5. Graphical illustration of the rationale behind the proposed approach.

until the final deadline and the remaining number rb of bytes to send. When the current throughput measurement T is above T_u or below T_l, the deadline margin is expected to grow above m_u or shrink to less than m_l before the end of the stream is reached. To avoid this, the difference m_δ between the current deadline margin and the approaching bound is calculated, as well as the remaining part $\alpha_\triangle$ of the range of α values that allows us to change the aggressiveness in the required direction (lines 11-20). Based on these values, the critical time period C by which leaving the margin bounds is expected, can be calculated (line 21). In combination with the remaining range $\alpha_\triangle$ and the time between consecutive parameter changes t_r, this critical time period is used calculate the degree to which the value of α should be changed, rounded to the change granularity α_g (lines 22-23).

To clarify the rationale behind the algorithm, an illustrative example is presented in Figure 5. In this example, we consider a video streaming session s with begin deadline d_0=30s, a total duration l_s=100s and a bit rate b_s=1Mbps. The presented sending curve and deadline curve respectively show the time when each byte is sent and when it is required. The streaming session started at 20s, the margin bounds are configured to m_u=20s and m_l=5s and the reactivity interval t_r is set to 1s. At time t=35s, 20Mbit has been sent already and the deadline margin amounts to about m=15s. At this point in time, rb=80Mbit remains to be sent in the next rt=95s. To be able to finish the stream without exceeding the margin bounds, the required throughput should be between $T_l = \frac{\text{80Mbit}}{\text{90s}} = \text{0.89Mbps}$ and $T_u = \frac{\text{80Mbit}}{\text{75s}} = \text{1.07Mbps}$. Given the current throughput measurement T=1.23Mbps and the current distance from the upper bound m_δ=5s, the deadline margin is expected to exceed the margin upper bound in $C = \frac{\text{5Mbit}}{\text{0.23Mbps}} = \text{21.74s}$. Given the reactivity interval t_r=1s, the current estimate results in 21 remaining chances to adapt α to reduce the aggressiveness. When currently $\alpha = 1.0$, α will be increased by $\alpha_\delta = \text{round}(\frac{(4.0-1.0)*1.0}{21.74}; 0.1) = 0.1$, resulting in α to be set to 1.1. When the difference between the current throughput and the required throughput would be higher, the change in α value would be more significant as the time to react would be shorter.

V. Evaluation

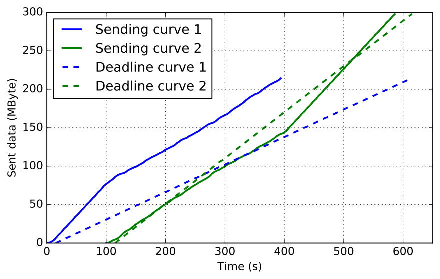
To evaluate the performance of the proposed approach, we first demonstrate the effectiveness of dynamically changing the AIMD parameters in a scenario with two sessions in Section V-A. Next, Section V-B will evaluate the proposed approach in larger scenarios using real-world characteristics. All simulations have been performed in NS-3 using a setup where all considered streams share a common bottleneck link on which RED queue management is applied.

A. Conceptual demonstration

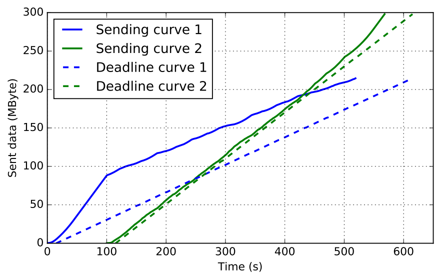
To show the benefits of using deadline-aware congestion control, a small-scale scenario is considered where two VoD streaming sessions share a bottleneck link with a capacity of 10Mbps and a delay of 30ms. The first stream has a bit rate of 3Mbps, has a begin deadline of 15s after the initial request and a total duration of 600s. After 100s, the second stream with a bit rate of 5Mbps is initiated with a begin deadline of 115s relative to the experiment start and a total duration of 500s.

Figure 6a shows the sending curves and the corresponding deadline curves of both video streams in a scenario where TCP New Reno congestion control is applied (α=1.0, β=0.5). It can be seen that during the first 100s, the first stream can use the full capacity of the link and can deliver all bytes long before their deadline. After 100s, the second stream starts and both streams get an equal share of the link capacity. However, this bandwidth share is not sufficient for the second stream to deliver its data before the deadline, resulting in deadline misses for over 70% of the data. In the worst case, the data is delivered only 50s after its deadline. However, the throughput for the first stream exceeds the required bit rate, causing the entire stream to be delivered about 200s earlier than the final deadline.

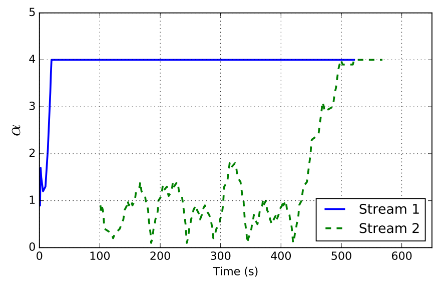
Figure 6b shows the performance of the proposed deadline-aware congestion control where the α parameter is dynamically changed for the same scenario, with the deadline margin bounds set to m_l=5s and m_u=20s. The resulting behavior of α is shown in Figure 6c. It can again be seen that during the initial 100s, the first stream can use the full capacity of the link and can deliver all bytes long before their respective deadlines. As the achieved throughput of that stream is significantly higher than the bit rate of the video, the α value quickly increases to its maximum value of 4.0 to become less aggressive and free up bandwidth for other streams if required. When the second stream starts, its fair share bandwidth part does not suffice to deliver the stream in time. Therefore, the second stream becomes more aggressive by lowering its α value. To keep the throughput within the predefined margin without using more bandwidth than required, α starts fluctuating between 0.1 and 1.5. When the first stream finishes, the second stream can use the entire capacity, resulting in a higher throughput than required. Therefore, the second stream backs off by increasing its α value. In this scenario, using deadline aware congestion control allows both streams to deliver all bytes in time.



(a) Performance of two VoD streaming clients using TCP New Reno congestion control.



(b) Performance of two VoD streaming clients using deadline-aware congestion control.



(c) Adaptive behavior of the value of α over time based on the current deadline information.

Figure 6. Conceptual demonstration of the proposed deadline-aware congestion control mechanism in a scenario with two VoD streaming sessions.

B. Larger scale evaluations

To evaluate the impact of the proposed approach in a scenario with multiple (deadline-aware) streams, several scenarios have been generated using realistic network traffic compositions. In each of these scenarios, the minimum bottleneck bandwidth, required to finish all streams without deadline misses, is established for both TCP New Reno congestion control and the deadline-aware congestion control. For this purpose, a binary search strategy has been applied to find the bottleneck bandwidth with a granularity of 100kbps. Given the probabilistic nature of the RED queue management, all simulations have been performed for 3 iterations. First, Section V-B1 considers a setup with only deadline-aware VoD streams, while more general setups, including deadline-aware live and VoD streaming sessions as well as non-deadline-aware traffic, are considered in Section V-B2.

1) VoD-only scenarios: To evaluate the performance of the proposed approach, VoD request traces have been constructed based on statistics provided by Conviva. In October 2015, Conviva opened access to viewer experience data, based on the analysis of 4 billion video streams per month spread across 180 countries[1]. According to this dataset, in the first quarter of 2016, the average bit rate of a VoD stream amounts to 2.4Mbps. Taking into account general adaptive streaming characteristics, each streaming session in the constructed request traces was assigned a uniformly distributed bit rate between 0.9Mbps and 3.9Mbps. The duration (in minutes) of each session can be modeled using a log-normal distribution with μ=2.2 and σ=1.5 [19], [20]. While the number of sessions is varied between 25 and 100, in each scenario the start times of the video streams are uniformly distributed over a period of 60minutes. In this way, different levels of load are considered. For each session, the sending of data can be started at most m_0 seconds before the begin deadline d_0 (i.e. at time $t = d_0 - m_0$). All flows share a bottleneck link with a delay of 30ms.

Multiple parameters have been evaluated, as presented in Table I. For each configuration, the approach has been evaluated in 10 scenarios, randomly generated according to the above characteristics. To assess the performance of the proposed approach, in each of the scenarios the minimal bottleneck bandwidth B_{da} that is required to deliver all streaming sessions without deadline misses using deadline-aware congestion control is defined by simulation and compared to the same bottleneck bandwidth B_{nr} required when using TCP New Reno congestion control. Furthermore, the theoretical lower bound for the bottleneck bandwidth B_{edf} when using the Earliest Deadline First (EDF) policy, scheduling the data with the earliest deadline amongst all flows at any point in time, is calculated. It is important to note that this optimal solution cannot be achieved in practice and does not take into account TCP overhead. Therefore, the actual lower bound will be significantly higher than B_{edf}, which serves as a benchmark. Based on these values, the metric μ is defined as shown in equation (5), representing the ratio between the achieved gain and the optimal achievable gain.

$$\mu = \frac{B_{nr} - B_{da}}{B_{nr} - B_{edf}} \tag{5}$$

The influence of the number of streaming sessions on the performance of the proposed deadline-aware congestion control mechanism, dynamically adapting the value of α, is presented in Figure 7 for different configurations of m_l and m_u. It can be seen that for each configuration of the deadline margin bounds, the performance of the proposed approach increases with the number of streaming sessions. As the contention is limited with fewer streams, the benefits of using deadline-aware congestion control are narrow. However, when the number of streaming sessions increases, the performance of the proposed approach significantly increases as well, saturating around μ=0.60. While the performance difference between the configurations is limited, m_l=5s and m_u=20s yields the best performance on average. When margin

Table I
EVALUATED PARAMETER CONFIGURATIONS.

Parameter	Values
Number of sessions	25, 50, 75, 100
m_l	0, 5, 10
m_u	10, 20
m_0	0.5, 5, 10, 20

[1]Conviva dataset - http://www.conviva.com/industry-data-portal/

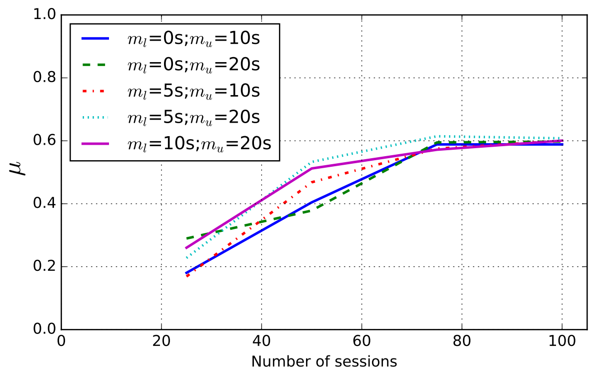


Figure 7. Performance of the deadline-aware congestion control mechanism by dynamically adapting the value of α for multiple parameter configurations in a VoD-only scenario.

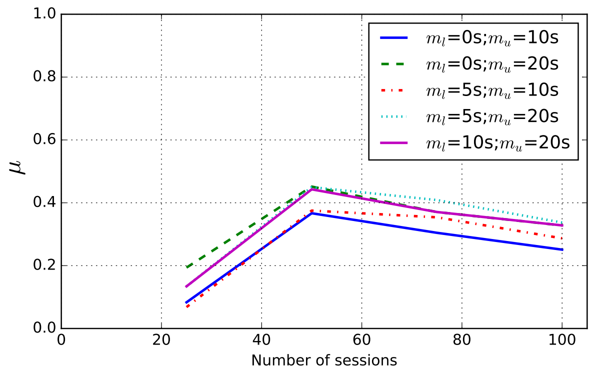


Figure 8. Performance of the deadline-aware congestion control mechanism by dynamically adapting the value of β for multiple parameter configurations in a VoD-only scenario.

bounds are set closer to the deadline, the reaction time for the algorithm increases, resulting in a slightly lower performance.

Similarly, Figure 8 shows the performance of the proposed deadline-aware congestion control mechanism dynamically adapting the value of β. It can be seen that for each configuration, the performance is lower compared to when dynamically adapting α. The explanation behind this finding is in the role of both parameters in the congestion control mechanism. While the value of α has an impact for every received acknowledgment, the value of β only impacts the behavior upon receiving triple duplicate acknowledgments in the event of congestion. Therefore, the impact of dynamically adapting β on the congestion window is perceived less frequently. Furthermore, the effect of changing the value of β cannot be perceived immediately, but only at the next congestion event. Based on this analysis, in the remainder of this paper the focus will be on the adaptation of the α parameter.

The results presented in Figure 7 show that for higher network loads, the proposed approach can yield around 60% of the theoretical upper bound for the achievable gain (i.e. μ=0.60) by dynamically adapting the value of α. To compare the performance of the deadline-aware congestion control with

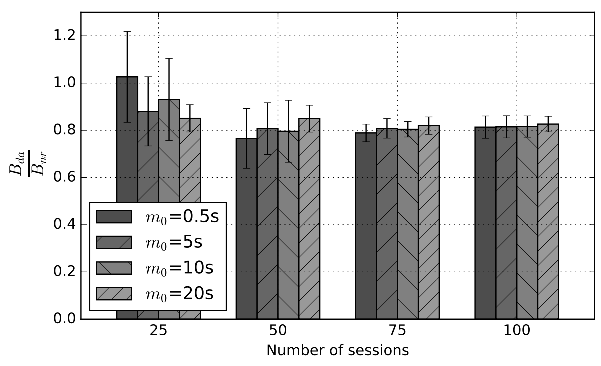


Figure 9. Relative performance of the deadline-aware congestion control by dynamically adapting α for m_l=5s, m_u=20s and multiple initial deadline margins m_0 in a VoD-only scenario.

the TCP New Reno congestion control, Figure 9 presents the ratio between the minimal bottleneck bandwidth required to streams all sessions without deadline misses, i.e. $\frac{B_{da}}{B_{nr}}$, and the corresponding standard deviation. Based on the above evaluation, the deadline margin bounds were set to m_l=5s and m_u=20s, considering multiple initial deadline margins m_0. It can be seen that for a low number of sessions, the minimal bottleneck bandwidth required to stream all sessions without deadline misses is 8% lower compared to using TCP New Reno congestion control on average, while for short initial deadline margins (m_0=0.5s) no gain can be achieved. For a higher number of sessions, the bottleneck bandwidth reduction amounts to between 15% and 23%. Furthermore, it can be seen that the influence of the initial deadline margin m_0 on the average performance is insignificant for a higher number of sessions. However, higher initial margins result in a more consistent performance increase, as presented by the decreasing standard deviations. In general, an average bottleneck bandwidth reduction of 16% is achieved.

2) General scenarios: In a general scenario, besides VoD streaming sessions, other types of traffic are present in the network as well. According to Cisco, in 2016 70% of all consumer Internet traffic consists of video streaming [1]. Out of the Conviva dataset it can be deduced that 36% of all streamed video data consists of live video, while the remaining 64% can be considered as VoD. Combining this information results in a traffic pattern where 44% of the network traffic consists of VoD, 26% consists of live streaming and the remaining 30% is considered as non deadline sensitive traffic. Given the size difference between the different types of streams, it is important to note that this division accounts to the amount of data streamed for each type, rather than the number of sessions for each type.

For the VoD sessions, the same characteristics as in the previous section have been used. To generate the live streaming sessions, a uniformly distributed bit rate between 1.1Mbps and 4.1Mbps has been used, based on the average bit rate of 2.6Mbps as reported by Conviva. Based on literature, the duration (in seconds) of these sessions is modeled using a log-

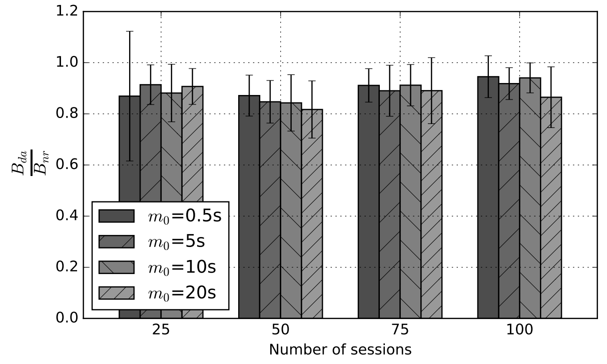


Figure 10. Relative performance of the deadline-aware congestion control by dynamically adapting α for m_l=5s, m_u=20s and multiple initial deadline margins m_0 in a general traffic scenario.

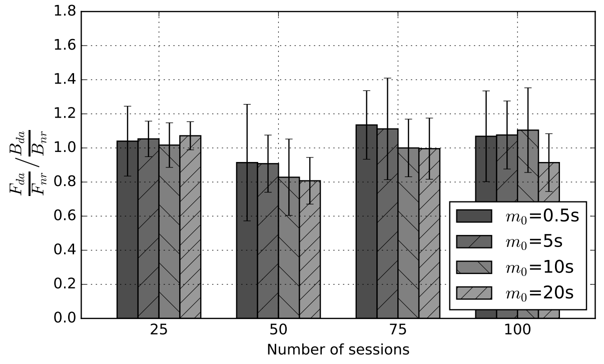


Figure 11. Impact of the deadline-aware congestion control mechanism by dynamically adapting α for m_l=5s, m_u=20s on the performance of non-deadline-aware traffic in a general traffic scenario.

normal distribution with μ=5.19 and σ=1.44 [20], [21]. As the playout of the live stream is considered to be 10s behind of the live signal, data can only be send at most 10s before its deadline. Therefore, the deadline margin m can never exceed 10s for live streaming sessions. For the non deadline sensitive traffic, regular file transfers with a uniformly distributed size between 0.5 and 600Mbyte have been generated.

Given the absence of deadline information for file transfers, the theoretical optimal gain when using EDF can not be defined. All non-deadline aware traffic has an infinite deadline, causing them to only be scheduled by EDF when no other deadline-aware traffic is present. As this does not allow a fair comparison, for the general scenarios the performance of the deadline-aware congestion control will be compared only to the TCP New Reno congestion control mechanism.

Figure 10 shows the relative performance of the deadline-aware congestion control compared to the TCP New Reno congestion control. Based on the results presented in Section V-B1, the value of α was dynamically adapted, using deadline margin bounds set to m_l=5s and m_u=20s. It can be seen that on average, a bandwidth reduction of between 5% and 18% can be achieved. As the file transfers cannot benefit from the deadline-awareness, the relative performance gain is lower compared to a VoD-only scenario. Furthermore, as the live streaming sessions can at most be 10s ahead of their deadlines, the potential benefits of deadline-aware congestion control are limited as well. However, even though only 44% of the traffic can fully take advantage of the deadline-aware congestion control, the minimal bottleneck bandwidth can be reduced by 11% on average compared to using TCP New Reno congestion control.

To analyze the impact of the deadline-aware congestion control strategy on the performance of regular TCP traffic, the notation F_{da} is introduced representing the sum of the throughput achieved by each of the regular file transfers in a scenario where deadline-aware congestion control is used. Similarly, the sum of the throughput of all regular file transfers when using TCP New Reno congestion control in the same scenario is denoted as F_{nr}. Based on these values, the ratio $\frac{F_{da}}{F_{nr}}$ shows the relative increase (> 1) or decrease (< 1) in throughput for non-deadline-aware traffic. In Figure 11, this value is compared to $\frac{B_{da}}{B_{nr}}$. In this graph, a ratio of 1 indicates that the change in throughput for the non-deadline-aware traffic is of the same relative magnitude as the change in bottleneck bandwidth. As a result, the relative bandwidth share of the non-deadline-aware traffic is unchanged. It can be seen that only in the scenarios with 50 sessions, the relative bandwidth share of the non-deadline-aware traffic is significantly reduced. In general however, with an average ratio of 1.00, the fairness with regular TCP traffic is maintained.

VI. Conclusions

In this paper, a deadline-aware congestion control strategy was presented based on the congestion avoidance phase of the TCP New Reno congestion control mechanism. By introducing deadline information at the transport layer, the modulation of the congestion window can be dynamically adapted to minimize the number of deadline-missing flows. The proposed approach only requires changes at the sender side and is fully transparent in the network. The deadline-aware congestion control mechanism has been thoroughly evaluated in both a VoD-only scenario and in co-existence with live streaming sessions and regular non-deadline-sensitive TCP traffic. It was shown that in a VoD scenario, the minimum bottleneck bandwidth required to finish all streaming sessions without deadline misses could be reduced by 16% on average. When considering more general scenarios, an average bottleneck reduction of 11% was achieved while maintaining a fair bandwidth share for regular TCP traffic.

Acknowledgment

M. Claeys and N. Bouten are funded by a grant of the Agency for Innovation and Entrepreneurship in Flanders (VLAIO). The research was performed partially within the ICON SHIFT-TV project (under grant agreement no. 140684). This work was partly funded by FLAMINGO, a Network of Excellence project (318488) supported by the European Commission under its Seventh Framework Programme.

References

[1] Cisco, "Cisco Visual Networking Index: Forecast and methodology, 2014-2019," Cisco, Tech. Rep., 2015.

[2] D. M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Computer Networks and ISDN Systems*, vol. 17, no. 1, pp. 1–14, 1989.

[3] N. Bansal and M. Harchol-Balter, "Analysis of SRPT scheduling," in *Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, vol. 29, 2001, pp. 279–290.

[4] K. Nichols, S. Blake, F. Baker, and D. Black, "Definition of the Differentiated Services field (DS field) in the IPv4 and IPv6 headers," *Network Working Group RFC 2474*, pp. 1–20, 1998.

[5] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowstron, "Better Never than Late: Meeting Deadlines in Datacenter Networks," in *Proceedings of the ACM International Conference on Communications Architectures, Protocols and Applications (SIGCOMM*, 2011, pp. 50–61.

[6] B. Vamanan, J. Hasan, and T. Vijaykumar, "Deadline-aware datacenter TCP (D2TCP)," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 4, pp. 115–126, 2012.

[7] G. Li, Y. Xu, and D. Cui, "A deadline and size aware TCP scheme for datacenter networks," in *Proceedings of the IEEE International Conference on Communication Technology (ICCT)*, 2013, pp. 366–371.

[8] R. Stewart, "Stream Control Transmission Protocol," *Network Working Group RFC 4960*, pp. 1–152, 2007.

[9] E. Kohler, M. Handley, S. Floyd, and J. Padhye, "Datagram Congestion Control Protocol (DCCP)," *Network Working Group RFC 4340*, pp. 1–130, 2006.

[10] K. De Schepper, B. De Vleeschauwer, C. Hawinkel, W. Van Leekwijck, J. Famaey, W. Van de Meerssche, and F. De Turck, "Shared Content Addressing Protocol (SCAP): Optimizing multimedia content distribution at the transport layer," in *Proceedings of the IEEE Network Operations and Management Symposium (NOMS)*, 2012, pp. 302–310.

[11] S. Liang and D. Cheriton, "TCP-RTM: Using TCP for real time applications," in *Proceedings of the IEEE International Conference on Network Protocols (ICNP)*, 2002, pp. 1–20.

[12] S. McQuistin, C. Perkins, and M. Fayed, "TCP goes to Hollywood," in *Proceedings of the ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, 2016, pp. 1–6.

[13] H. P. Shiang and M. Van Der Schaar, "A quality-centric TCP-friendly congestion control for multimedia transmission," *IEEE Transactions on Multimedia*, vol. 14, no. 3, pp. 896–909, 2012.

[14] C.-Y. Hong, M. Caesar, and P. B. Godfrey, "Finishing flows quickly with preemptive scheduling," in *Proceedings of the ACM International Conference on Applications, technologies, architectures, and protocols for computer communications*, 2012, pp. 127–138.

[15] A. Munir, I. A. Qazi, Z. A. Uzmi, A. Mushtaq, S. N. Ismail, M. S. Iqbal, and B. Khan, "Minimizing flow completion times in data centers," in *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2013, pp. 2157–2165.

[16] T. Henderson, S. Floyd, A. Gurtov, and Y. Nishida, "The New Reno modification to TCP's fast recovery algorithm," *Internet Engineering Task Force RFC 6582*, pp. 1–16, 2012.

[17] M. Allman, V. Paxson, and W. Stevens, "TCP congestion control," *Network Working Group RFC 2581*, pp. 1–14, 1999.

[18] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, 1993.

[19] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng, "Understanding user behavior in large-scale video-on-demand systems," *ACM SIGOPS Operating Systems Review*, vol. 40, no. 4, pp. 333–344, 2006.

[20] I. Ullah, G. Doyen, G. Bonnet, and D. Gaïti, "A survey and synthesis of user behavior measurements in P2P streaming systems," *IEEE Communications Surveys and Tutorials*, vol. 14, no. 3, pp. 734–749, 2012.

[21] E. Veloso, V. Almeida, W. Meira, A. Bestavros, and S. Jin, "A hierarchical characterization of a live streaming media workload," *IEEE/ACM Transactions on Networking*, vol. 14, no. 1, pp. 133–146, 2006.