Performance and Improvements of TCP CUBIC in Low-Delay Cellular Networks

1st Philipp Bruhn
Ericsson Research
Ericsson GmbH
Herzogenrath, Germany
philipp.bruhn@ericsson.com

2nd Mirja Kuehlewind
Ericsson Research
Ericsson GmbH
Herzogenrath, Germany
mirja.kuehlewind@ericsson.com

3rd Maciej Muehleisen
Ericsson Research
Ericsson GmbH
Herzogenrath, Germany
maciej.muehleisen@ericsson.com

***Abstract*—Internet usage is still growing every day and there is an even stronger increase in use of mobile devices on mobile networks to access the web. Ensuring good performance in cellular networks for web traffic, which today often consists of many short-lived TCP flows, is therefore particularly important. For TCP, and likely in the future also for QUIC, CUBIC is the most widely deployed congestion control algorithm and in Linux by default used with HyStart. In this paper we investigate performance impairments that especially happen when the combination of CUBIC and HyStart is used in low-delay cellular networks. To explain the problem in detail, we show measurements of characteristic per-packet round-trip times present in mobile networks due to cellular uplink scheduling. Inherent round-trip time variations often trigger an early exit of Slow Start, causing CUBIC to underutilize the available capacity. Further, we have implemented two straightforward HyStart fixes that enhance the performance of CUBIC significantly. In our LTE test bed, the mean throughput of a 5 MB data transfer increases by 10.5 %, and the page loading times decrease respectively. Given the nature of web traffic today, these fixes can improve the web experience of users noticeably, which also shows the importance of measurements, testing, and subsequent optimizations when designing and evaluating transport and congestion control mechanisms, especially for mobile networks.**

***Index Terms*—Mobile network, TCP CUBIC, HyStart**

I. Introduction

Currently TCP is the most widely used transport protocol in the Internet [1]. Many common application layer protocols, e.g. Hypertext Transfer Protocol (HTTP), rely on TCP for reliable data transmission. For TCP, and likely in future also for QUIC [2], the most widely deployed congestion control algorithm is CUBIC [3], [4]. The CUBIC implementation in Linux uses HyStart [5] as the default Slow Start algorithm. HyStart is a hybrid Slow Start variant that relies on delay-based semantics to avoid the so-called Slow Start overshoot, i.e. exceeding the available capacity leading to burst packet losses. However, when the combination of CUBIC and HyStart is used in cellular networks, HyStart often causes a non-optimal behavior by triggering a premature exit of the Slow Start phase. CUBIC then respectively enters the Congestion Avoidance phase way below the maximum capacity and subsequently requires a longer time to reach maximum throughput. This effect has been observed in earlier studies [6] but was not investigated systematically.

In this paper we present measurements taken in a lab LTE test bed resembling an edge computing scenario, which is believed to become increasingly important in the future and considered to be specifically challenging as delay variations have a high impact. Our measurements explain in more detail the root cause of this problem, namely the inherent delay variations in cellular networks. Delay variations partly arise since TCP ACKs are sent in the uplink in response to scheduling grants for the shared resources obtained by different request procedures, which results in very different Round-Trip Times (RTTs) for individual data packets. In this context we discuss the rarely considered per-packet RTT at the beginning of a TCP connection.

We then show that this early transition to Congestion Avoidance results in significant underutilization of the cellular network capacity, particularly for short-lived CUBIC flows. Roughly 50 % of the web traffic worldwide is associated with mobile devices [7]. Moreover, as a majority of the web servers use Unix-like operating systems, i.e. Linux [8], and web traffic typically consists of many small-sized pages [9], short-lived CUBIC flows play a significant role on the web and the detected problem can strongly impair the web experience of users. We show that even in a controlled lab environment, where no external influences are causing throughput degradation, the average throughput of individual short CUBIC flows varies significantly. This effect is noticeable and impairs user performance even in scenarios with less stable network load or signal quality due to cross traffic or mobility, as we have verified in drive tests. However, in this paper we discuss our results obtained in the controlled lab environment to separate the effect of premature Slow Start exit from other impairments. Our results show that the performance in terms of throughput substantially depends on the Congestion Window (CWND) size at the Slow Start exit point, as set by HyStart.

We demonstrate that two changes of the HyStart implementation improve the performance of TCP CUBIC significantly, especially for small data transmissions. For one, our investigations led to the detection of a bug in the Linux kernel that was inconsistent with the original HyStart proposal in [5]. The other change considers a case where the minimum (smallest

ISBN 978-3-903176-48-5

observed) RTT decreases, which is unusual in fixed networks but has a huge impact in our scenario. This was fixed in Linux kernel version 5.7 and back-ported to all current longterm-support versions based on our inputs.

As such this paper makes the following contributions:

- A detailed analysis of the per-packet RTT at the beginning of TCP flows as influenced by cellular uplink scheduling
- A performance evaluation of short-lived TCP CUBIC flows in our LTE test bed showing the non-optimal behavior of TCP CUBIC with HyStart in cellular networks
- A significant performance improvement for short-lived TCP CUBIC flows based on two small fixes of the HyStart implementation, namely one bug fix addressing a wrong selection of the HyStart sampling phase causing to a significant impact in mobile networks, and a small HyStart algorithm change to consider variable per-packet RTTs more generally, both showing the importance of testing congestion control algorithms in mobile networks.

In the following section we first discuss similar findings from related work as well as needed background information and references. In Sec. III we present our test bed setup. In Sec. IV we elaborate on delay and throughput variance observed in the test bed due to the cellular uplink scheduling and discuss the behavior and improvements of HyStart when used in cellular networks. Finally, we draw a conclusion about the findings of our work.

II. Background and Related Work

A. Delay variations in mobile networks

There is a large portion of work that investigates and measures the delay variations in mobile networks and, in particular, in cellular network [10]. Most of the papers concentrate on the effect of jitter for long-lived TCP transmissions causing spurious timeouts [11]–[13]. Huang et al. [14] are also looking into the performance of short-lived flows, acknowledging that predominance of such flows in the web and the problem of early Slow Start exit. Similar as other studies [15], [16] they notice the problem of premature Slow Start exit but do not study the root cause of this problem in detail. In this paper we agree with their conclusion that the development of transport mechanisms must put a stronger focus on characteristics of and consequent challenges in cellular networks. We therefore specifically look into the behavior and performance of TCP CUBIC with HyStart, which is the default setting in Linux and thus widely deployed.

B. HyStart and TCP CUBIC

HyStart [5] is a hybrid Slow Start algorithm that is designed to avoid overshooting the available capacity and as such prevents burst packet losses during Slow Start. HyStart uses two techniques to find a suitable Slow Start exit point: One mechanism is based on the temporal spacing of ACKs and the ACK train length, but is not relevant for the findings in this paper. The other mechanism looks at the RTTs of individual packets. As per current[1] Linux implementation [18], which differs from the original proposal given in [5], the latter can be summarized as follows:

During Slow Start the data transfer happens in rounds. If the initial CWND size $W_{\mathrm{I}} = 10$,[2] the sender transmits 10 full-sized packets in the first round, 20 full-sized packets in the second round and so on. For each round, HyStart identifies the current RTT as the smallest RTT sample based on the first eight ACKs of the round. These ACKs form a sampling phase in which HyStart never leaves the Slow Start. After this sampling phase, HyStart triggers the exit of the Slow Start if the current RTT exceeds the overall minimum RTT plus a threshold, which again depends on the minimum RTT.

As HyStart transitions to Congestion Avoidance the current CWND size is used as $W_{\max}$, which is the inflection point of the cubic curve, and is usually set to the CWND size just before the last packet loss. Thus, the cubic function starts at its inflection point, which is a region where the CWND grows most slowly. To always achieves at least the throughput of Reno [19], CUBIC defines a so-called TCP-friendly region. In this region, the current[1] CUBIC implementation in Linux stipulates a window growth of 5 % per RTT, until the window growth intended by the cubic function exceeds those 5 %. At that point the TCP-friendly region is left and after that the cubic function defines the window growth.

Since the transition from Slow Start to Congestion Avoidance reduces the CWND growth from 100 % per RTT to 5 % per RTT, a premature transition strongly impairs the throughput in the initial ramp-up period. This effect has been observed in related work [6], [20]. A comparison of the behavior of several TCP variants over LTE shows that, at beginning of a flow, CUBIC is less aggressive than the others and takes longer to reach maximum throughput [6]. Without any further details, this effect is explained by a premature transition to Congestion Avoidance triggered by the delay-based HyStart algorithm. According to [20], the HyStart algorithm does not function well in cellular networks due to delay jitter that occur independent of network congestion while HyStart assumes all delay arises from queuing and is thus a sign of network congestion. In this paper we further investigate this problem and show specifics of the cellular scheduling are making this problem particularly important.

HyStart++ [21] is a new Slow Start algorithm used by Microsoft to address the early Slow Start exit issue of HyStart. It introduces another phase based on the Limited Slow Start algorithm [22] that kicks in after HyStart triggers the exit of Slow Start. This circumvents the slow window growth and thus the slow throughput increase of TCP CUBIC in the TCP-friendly region, but does not address the actual root cause of the issue. As there is no HyStart++ implementation for Linux available yet, we were not able to test it in our setup. We expect our fixes to limit the premature Slow Start exit issue and, together with HyStart++ enhancements, to improve the performance even further.

[1]Linux kernel v4.19, but the TCP CUBIC implementation did not change in later versions.

[2]Default in Linux kernel v4.19 and later versions.

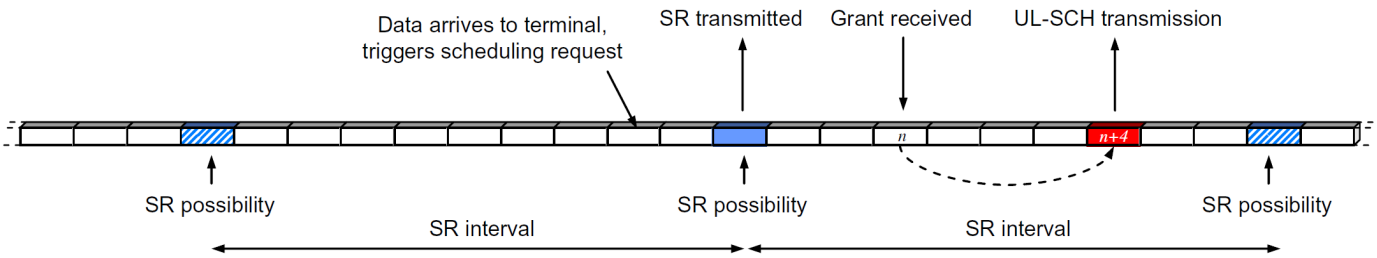


Fig. 1: SR procedure. Reproduced from [17].

C. LTE Uplink Scheduling

In cellular, e.g. LTE, networks the shared uplink resources are dynamically allocated by the eNodeB (eNB) scheduler on a per subframe basis, i.e. for each 1 ms-long Transmission Time Interval (TTI) in LTE [17]. The same principle is also used in 5G networks, but with a flexible TTI duration, which may also be less than 1 ms. Thus, the User Equipment (UE) requires a scheduling grant from the eNB, telling the UE which time-frequency resources to use in order to transmit any data on the Uplink Shared Channel. Since the scheduler cannot know if the UE has data in a buffer, the UE must signal its resource needs to the eNB, which may be done in two different ways.

If the UE has a valid uplink grant, it may piggyback a Buffer Status Report (BSR) onto a data transmission using this grant. The BSR tells the scheduler roughly how much more data the UE has in a buffer waiting to be scheduled. Otherwise, if the UE does not have any uplink grant, it must transmit a Scheduling Request (SR). In this case, the scheduler only knows that data is being buffered, but not how much, so that it typically grants small amount of uplink resources.

Fig. 1 schematically depicts the SR procedure. Due to processing and scheduling delays, the eNB usually sends an uplink grant 4 ms after receiving the SR. Besides, if the UE receives the uplink grant in subframe n, it always sends in subframe $n + 4$. If data arrives at the UE and triggers a SR, it typically takes 8 ms plus the waiting time for the SR possibility before the UE can send the data. In a customary cell configuration, a SR possibility comes every 10 ms [23]. After sending a BSR to the eNB, it takes further 8 ms until an UE can use a therewith received uplink grant.

As such, some data is sent on the grant triggered by the SR while other data is sent on the grant triggered by the BSR. This causes variations in the delay in the uplink, which consequently leads to variations in the RTT observed for individual packets of a TCP connection. As shown in Sec. IV-D, especially measuring a delay decrease outside of the HyStart sampling phase can lead to a wrong detection of a delay increase and thereby result in a premature exit from the Slow Start phase.

III. Test Bed Setup

Fig. 2 shows our test bed setup consisting of a PC shown on the left side and a lab LTE network deployed in a rack shown on the right side. The eNB, comprising Baseband Unit (BBU) and Remote Radio Head (RRH), implements LTE

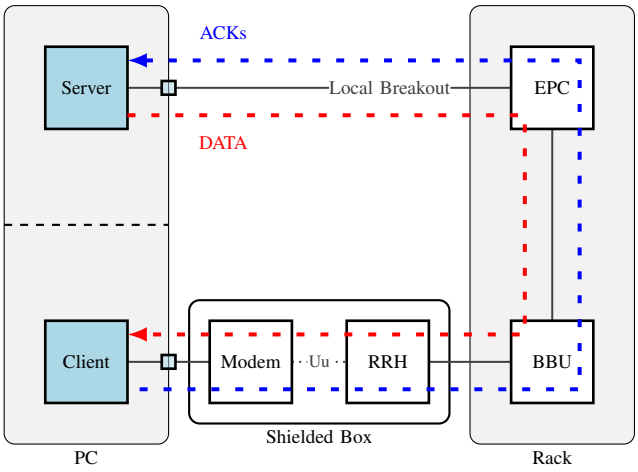


Fig. 2: Test bed setup.

uplink scheduling with default configuration. To guarantee consistent radio conditions, RRH and LTE modem (connected to the PC) are placed inside a shielded box. The aim is to ensure a controlled environment and a good reproducibility of the results. Nevertheless, related drive tests indicate that the effects explained in this paper also occur under varying radio conditions, other cell configurations, including 5G, as well as scenarios with higher end-to-end delay (to a remote server), due to the fact that the highly varying per-packet RTT is the dominate factor adversely affecting the HyStart behavior. We note that cellular networks use separate (and typically large) queues for each UE and fairness schemes for radio resource scheduling, which explains why the effects discussed in this paper also appear in the real world, e.g., in case of radio resource contention (in case of more UEs).

The PC is a IPU675 (Intel Processing Unit) single-board computer from NRG Systems [24] that is connected via the USB standard to the AirPrime MC7430 M2M Module from Sierra Wireless [25], a LTE Cat6 modem. The modem supports a variety of frequency bands, a peak download rate of 300 Mbps and a peak upload rate of 50 Mbps.

The eNB operate on LTE Band 7 with a bandwidth of 10 MHz, i.e. 50 Physical Resource Blocks (PRBs), per direction. It utilizes a 2x2 Multiple Input Multiple Output (MIMO) scheme and a Modulation and Coding Scheme (MCS) with a 64-Quadrature Amplitude Modulation (QAM) in the Downlink (DL) and a 16-QAM in the uplink. Using *iPerf3* [26], we

obtained a maximum throughput of $L_{\mathrm{DL}} = 60.3\,\mathrm{Mbps}$ in the DL and $L_{\mathrm{UL}} = 25.6\,\mathrm{Mbps}$ in the uplink.

As the PC has a powerful CPU and a large RAM, we run both the HTTP server and the client, which triggers downloads from the server, on the same PC. Each process operates in different network namespaces, which are duplicates of the network stack that have their own network devices, routing rules and a logically separated share of the kernel resources [27], [28]. This is used to force all data traffic to pass through the LTE network instead of being routed locally.

For each download, we measure the transfer duration from the point of view of the client, i.e. at the application level. In detail, the client sends a HTTP request to download a test file that triggers a HTTP response carrying the file as payload. The transfer duration includes the time required for the TCP connection establishment as well as the transmission of the HTTP request and response messages. Hence, it corresponds to the total time that the client must wait for the data transfer to be completed. Further, we use *Tcpdump* to measure the RTT of individual packets.

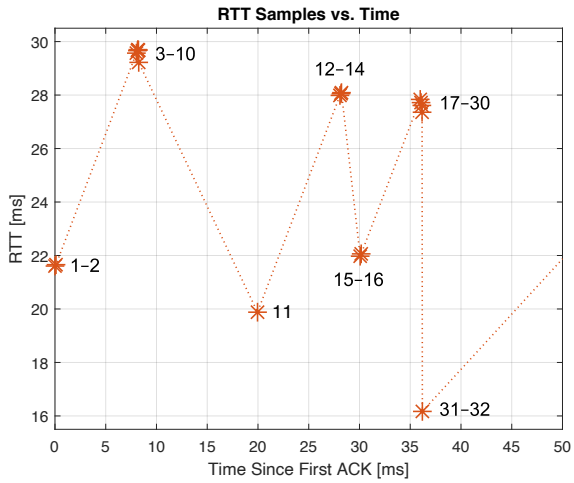
IV. Analysis of HyStart in Cellular Networks

We first take a deeper look at the RTT variance in our test bed at the beginning of a data transfer and spell out the role of cellular uplink scheduling for the premature Slow Start exit. We then we look more closely at the behavior, problems, and improvements of TCP CUBIC and HyStart in cellular networks. We show a problematically high throughput variance for short-lived flows and explain how this throughput variance arises from different Slow Start exit points set by HyStart. We further show that the root cause of this observation lies in an unexpected and unintended behavior of HyStart when drops of the RTT due to cellular uplink scheduling are measured. Finally, we show how two small fixes of the HyStart implementation in the Linux kernel enhance the performance of TCP CUBIC noticeably, demonstrating how important testing and optimization are for mobile networks.

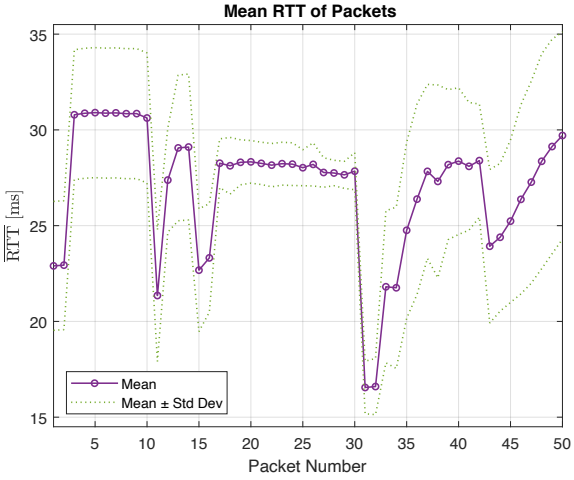
A. Delay Variations due to Cellular Uplink Scheduling

Fig. 3 shows the per-packet RTT measurements at the beginning of a TCP flow for a downlink data transfer: Fig. 3a shows an example of the first 32 RTT samples of a particular transfer plotted against the elapsed time since the sender received the first ACK, while Fig. 3b displays the mean RTT of individual packets based on 2×10^4 transfers, which shows that this is a typical pattern. While setup follow customary cell configurations, we have observed a similar pattern for other configuration, but the exact RTT timing may differ.

Looking more closely at Fig. 3a, it can be seen that the first two packets have a smaller RTT than the other eight packets of the initial window. However, all packets have been sent at the same time and were received almost at the same time, because the radio conditions in the shielded box are always good, so that the delay variance in the DL, e.g. due to retransmissions, is small [23]. With a data rate of $L \approx 60\,\mathrm{Mbps}$ in the DL, the first ten packets with a data volume of $15\,\mathrm{kB}$ can be



(a) Example of RTT samples vs. time.



(b) Mean RTT of individual packets.

Fig. 3: Per-packet RTT measurements in test bed.

sent in $2\,\mathrm{ms}$, i.e. two TTIs. Confirmed by measurements, they generally arrive at the receiver with a temporal separation of up to $1\,\mathrm{ms}$. The receiver then generates and transmits all ten ACKs to the modem more or less simultaneously. However, if the modem must obtain a grant via SR, such a grant typically provides only a small amount of uplink resources [23]. The modem then uses it to send two ACKs and a BSR. The other eight ACKs have to be buffered until a grant in response to the BSR is received.

This effect accumulates over time, since ACKs received in batches trigger new data packets to be sent in batches, and two batches of ACKs for packets of the initial window lead to four batches of ACKs for packets in the second round. This sums up to the point where the first ACKs of the third round (ACKs for packets $31 - 32$) are sent together with the last ACKs of the second round (ACKs for packets $17 - 30$). We note that packets $31 - 32$ are sent in response to receiving the ACK for packet 11, while packets $17 - 30$ are sent in response to receiving the ACKs for packets $4 - 10$, therefore packets

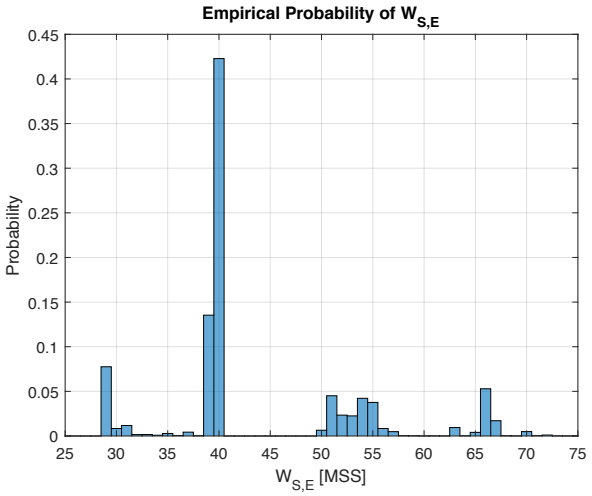


Fig. 4: Probability of CWND size at Slow Start exit point in test bed.

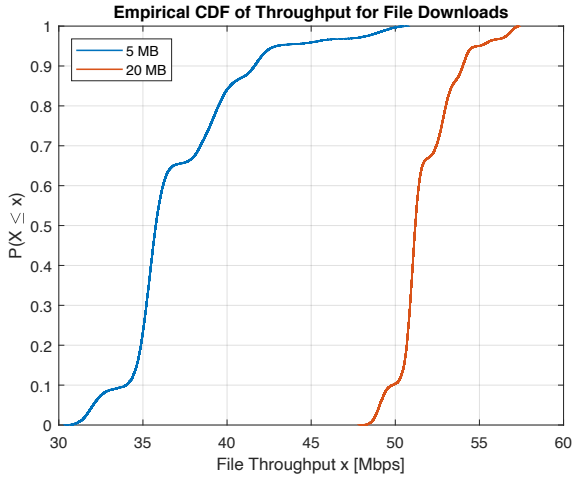


Fig. 5: Empirical CDFs of file throughput for 5 MB and 20 MB file downloads.

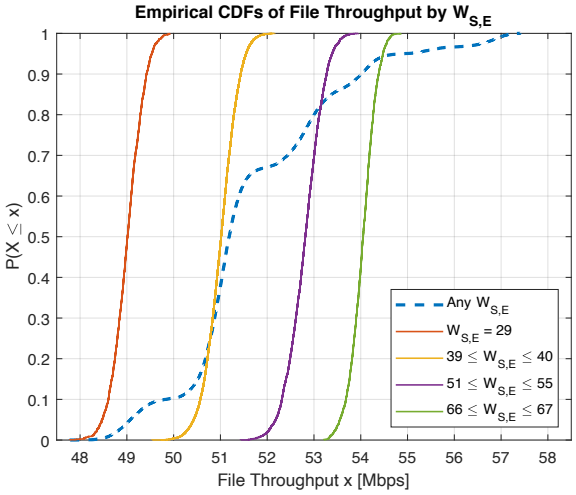


Fig. 6: Empirical CDFs of file throughput by CWND size at Slow Start exit point for 20 MB file downloads.

$31-32$ are sent roughly $12\,\mathrm{ms}$ later than packets $17-30$. The first ACKs of the third round do not have to wait for their own uplink grant, so that they are sent (and consequently received) together with the last ACKs of the second round. This leads to a misleading drop of the RTT seen for packet 31 and 32, due to their later transmission. This drop and the subsequently observed increase of the RTT is misinterpreted by HyStart as increase in queuing delay and leads to an early exit of Slow Start.

B. Early Slow Start Exit

In the following we further analyse the issue of early Slow Start exits. With the test bed shown in Fig. 2, we performed 2×10^4 downloads of a $20\,\mathrm{MB}$ file and measured the CWND size at the Slow Start exit point, $W_{\mathrm{S,E}}$, for each TCP flow. Fig. 4 depicts the empirical probability of $W_{\mathrm{S,E}}$.

Although the radio conditions are stable, the experiment still contains a source of randomness due to different time offsets between subframe boundary and (randomized) transmission begin. This leads to some variations in the measured per-packet RTTs for each run and, as a consequence, to different Slow Start exit points, as described in detail in Sec. IV-D.

According to Fig. 4, in roughly $56\,\%$ of cases, the Slow Start ends at a CWND size of 39 or $40\,\mathrm{MSS}$. In another $39\,\%$ of cases, it ends at a CWND size below $75\,\mathrm{MSS}$, but not on 39 or $40\,\mathrm{MSS}$. Neglected by Fig. 4, in about $5\,\%$ of cases, it ends at a CWND size above $75\,\mathrm{MSS}$ (to a maximum of $390\,\mathrm{MSS}$). The latter cases occurred too rarely to make sound statements or draw conclusions. The results indicate that the CWND size at the Slow Start exit point takes on very diverse values.

In our test bed, we observed a maximum throughput of $L = 60.3\,\mathrm{Mbps}$, an average base RTT of $28\,\mathrm{ms}$ and a Maximum Segment Size (MSS) of $1348\,\mathrm{bytes}$. Based on these values, the Bandwidth-Delay Product (BDP) of the communication path between server and client is around $157\,\mathrm{MSS}$. Hence, in almost all of the cases, the CWND size at the Slow Start exit point is far below the BDP.

C. Throughput Variance

In this section we analyze the throughput variance which can be observed for individual TCP CUBIC connections in cellular networks due to HyStart's non-optimal exits of Slow Start. Using the test bed shown in Fig. 2, we measured the transfer duration of 2×10^4 file downloads and calculated the average file throughput as file size divided by transfer duration. Fig. 5 depicts the empirical Cumulative Distribution Function (CDF) of the throughput achieved by downloads of $5\,\mathrm{MB}$ and $20\,\mathrm{MB}$ files.

As expected, a smaller file achieves a much lower throughput compared to a larger file. This is because a connection for transferring a small file spends most time in throughput ramp-up, or never even reaches maximum data rate.

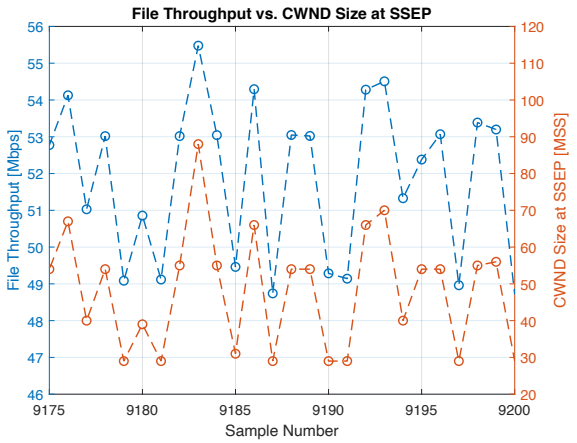


Fig. 7: File throughput vs. CWND size at Slow Start exit point for 20 MB file downloads.

However, a more interesting observation is that the average throughput achieved by individual file transfers varies significantly, for both file sizes. This appears to be striking, as both experiments have a single client without interference from cross traffic and ideal radio conditions[3]. Further, it can be seen that the experiments with the smaller file shows a much higher throughput variance than the experiments with the larger file.

This variance is closely linked to the variation of $W_{\mathrm{S,E}}$, as it can be seen in Fig. 6, which again depicts the empirical CDFs of throughput for 20 MB transfers but separately by CWND size at Slow Start exit point, $W_{\mathrm{S,E}}$. The results come from the same experiment but are sorted based on the measured value of $W_{\mathrm{S,E}}$. The chosen ranges of $W_{\mathrm{S,E}}$ are based on the clusters seen in Fig. 4. It can be observed that, for each of those ranges, the variance is much lower than for the overall throughput curve and roughly equally distributed.

The strong correlation between the average throughput achieved by a connection against the CWND size at the Slow Start exit point $W_{\mathrm{S,E}}$ of the respective connection can also be observed in Fig. 7. In fact, the Pearson correlation coefficient was calculated as $86.25\,\%$ with a p-value of nearly zero.

D. HyStart Behavior

In the previous section we have shown that the Slow Start exit point is the main cause of the throughput variance. In this section we will discuss in detail why HyStart causes this behavior in cellular networks. In our test bed the maximum throughput is reached at a CWND size of about 157 MSS, which is the BDP of the communication path between server and client. However, on average, HyStart leaves Slow Start at a CWND size of 48 MSS (cf. Fig. 4).

As explained in Sec. II-B, HyStart relies on two mechanisms to find a suitable Slow Start exit point before the CWND size exceeds the path capacity. In our test bed, using *Netstat* [29], we can see that HyStart never triggers based on the length of the ACK train, but instead always on the increase of the RTT (cf. Sec. II-B). Thus, Slow Start ends when the following comes true:

$$\begin{aligned} &\mathrm{RTT}_{\mathrm{cur}} > \mathrm{RTT}_{\mathrm{min}} + \xi\,, \quad \text{where} \\ &\xi = \max\left(4\,\mathrm{ms}, \min\left(\mathrm{RTT}_{\mathrm{min}}/8, 16\,\mathrm{ms}\right)\right)\,, \end{aligned} \tag{1}$$

and $\mathrm{RTT}_{\mathrm{min}}$ is the overall minimum RTT of the connection. $\mathrm{RTT}_{\mathrm{cur}}$ is the current RTT of the connection, which is set on a per-round basis to the smallest RTT sample of the first eight ACKs of the transmission round (cf. Sec. II-B).

In the test bed $\mathrm{RTT}_{\mathrm{min}}$ is always below 32 ms and therefore $\xi = 4\,\mathrm{ms}$ at all times. The fact that ξ is always set to the smallest possible value raises the suspicion that the delay increase mechanism is prone to false positive detection of network congestion.

Looking at (1), HyStart may trigger for two different reasons: First, as originally intended for fixed networks, it triggers when the current RTT increases above the critical value of $\mathrm{RTT}_{\mathrm{min}} + \xi$. However, it may also trigger if the overall minimum RTT decreases below the critical value of $\mathrm{RTT}_{\mathrm{cur}} - \xi$. While in the former case Slow Start is left after the HyStart sampling phase, in the latter case, which is seen as an error case in a fixed network with a constant base RTT, Slow Start is left immediately.

We evaluated the empirical probability of the two reasons due to which the delay increase mechanism actually triggers. In about $30\,\%$ of the cases, HyStart triggers on the increase of the RTT. However, in about $70\,\%$ of the cases, it triggers due to a misleadingly observed decrease of the RTT. In both cases, the delay detection mechanism does not work as intended, because it ends Slow Start way too early.

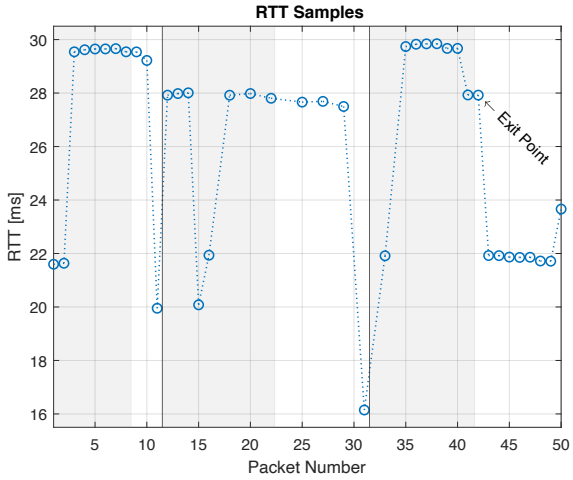
Fig. 8 shows example series of per-packet RTT values from a measured connection for each of the two cases due to which the Slow Start exit can be triggered. The gray areas show the sampling phase of each transmission round and the vertical lines the points at which HyStart resets $\mathrm{RTT}_{\mathrm{cur}}$.

In Fig. 8a, HyStart ends Slow Start after the sampling phase of the third round, because $\mathrm{RTT}_{\mathrm{cur}}$, which is minimum of the eight values in the gray sampling phase, increases from $\mathrm{RTT}_{\mathrm{cur},2} \approx 20\,\mathrm{ms}$ in the second round to $\mathrm{RTT}_{\mathrm{cur},3} \approx 22\,\mathrm{ms}$ in the third round. As $22\,\mathrm{ms} > 16\,\mathrm{ms} + 4\,\mathrm{ms}$, and with $\mathrm{RTT}_{\mathrm{min}} \approx 16\,\mathrm{ms}$, (1) comes true and the delay increase mechanism triggers on the first ACK after the sampling phase. At this point the CWND size is only 51 MSS and far below the BDP.

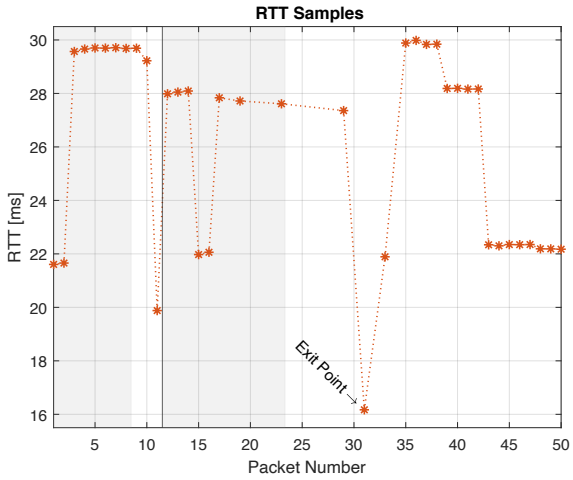
In Fig. 8b, $\mathrm{RTT}_{\mathrm{cur},2}$ is at about 22 ms and as such does not cause the early exit. However, since it is larger in this case, the delay increase mechanism triggers when a new $\mathrm{RTT}_{\mathrm{min}}$ is measured on the last packet of the second round at an even smaller CWND size of 40 MSS.

Fig. 8 clearly demonstrates the shortcomings of HyStart for cellular networks, which have typical RTT variations at the transmission start. Although the two examples only differ in the one RTT sample for the fifteenth packet, the Slow Start exit point $W_{\mathrm{S,E}}$ differs by more than $20\,\%$ and, in both cases, HyStart significantly underestimates the available capacity.

[3]High signal quality, no interference.

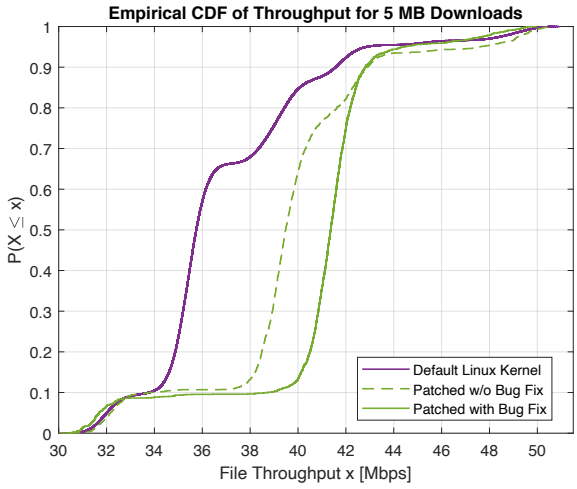


(a) Reason 1: RTT increases.

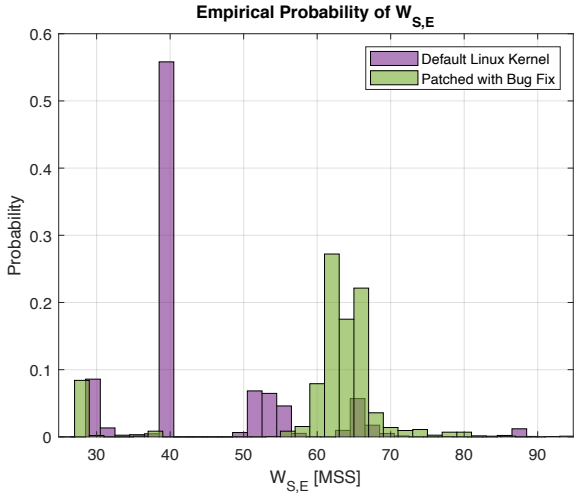


(b) Reason 2: RTT decreases.

Fig. 8: Examples of the reasons why delay increase mechanism triggers.



(a) Empirical CDF of file throughput.



(b) Probability of CWND size at Slow Start exit point.

Fig. 9: Comparison of measurement results for default and patched kernel.

HyStart misinterprets the variations of the base delay in cellular network as network load. Even worse, it reacts to an unexpected decrease in the base RTT by immediately leaving the Slow Start phase (way too early). This behavior has a huge impact on the performance of short-lived flows, since CUBIC catches up only slowly in its TCP-friendly region. In the following section we explain how the HyStart behavior can be improved by two simple tweaks and further discuss how HyStart as well as potentially other hybrid Slow Start algorithms can be enhanced to be more robust to base delay variations on cellular network.

E. HyStart Improvements

In this section we show that we were able to significantly improve the performance of CUBIC based on two rather small fixes of the HyStart implementation.

The first fix is a bug that was not intended in the HyStart algorithm. Effectively, the notion of the starting point of a round in the Linux implementation is off by one ACK such that the $\mathrm{RTT}_{\mathrm{cur}}$ reset occurs one ACK too late. Hence, HyStart resets the current RTT after the first ACK of a new round, not after the last ACK of the old round. Consequently, the first ACK of a round cannot affect the estimation of current RTT. In fixed networks, where the base RTT is assumed to be mostly constant, this does not have a big impact, but in mobile network, this first ACK, sent after a short idle period, typically has a smaller delay and is one of the main causes for premature Slow Start exit.

The second improvement addresses the problem that the delay detection mechanism can also be also incorrectly triggered by a decrease of the RTT. The original algorithm design does not anticipate the possibility of a RTT drop outside of the sampling phase of a round. It presumes that the minimum

RTT is always found inside the sampling phase.

Our work led to two patches that noticeably improve the behavior and therefore the performance of CUBIC in the test bed. One patch fixes the bug concerning starting point of a transmission round in HyStart, just by adding a "-1" at the right place. The other patch enables any RTT sample to reduce the current RTT, even if a new minimum RTT is observed outside of the sampling phase of a round, which prevents a spurious Slow Start exit point upon a drop of the RTT. As it can be seen in Fig. 9a, these two changes (green solid line) improve the mean file throughput for the $5\,\mathrm{MB}$ downloads by about $10.5\,\%$, while the median improves by about $15.7\,\%$. Moreover, the file throughput variance decreases and the portion of samples which are close to the mean increases from $56\,\%$ to $84\,\%$.

As another data point, the mean throughput of a $20\,\mathrm{MB}$ data transfer (not shown in Fig. 9) still increases by 3.6 %, and the median gains 5.1 %. Naturally, we observe a smaller relative gain for longer data transfer since HyStart only affects the sending rate in the beginning of a TCP connection until the maximum throughput is reached. For small data transfers, however, which can be predominant in web scenarios, the available capacity is often not fully utilized, so that high relative performance gains by the two changes can be observed.

The insights about the implementation flaw and unfavorable behavior of HyStart were reported to the developers of CUBIC in the Linux kernel and a change to correctly consider RTT decreases was integrated into main Linux kernel with version 5.7 and back-ported to the current longterm-support versions.

The green dashed line in Fig. 9a shows the gain with only the patch allowing updates of the current RTT outside the sampling period. An equivalent algorithmic change has already been deployed in recent Linux kernel releases based on our input. This small but important change covers some cases of early Slow Start exit but not all. The bug fix further improves the performance of CUBIC with HyStart, namely by covering more cases of early exit, e.g., the case shown in Fig. 8a. Even though this fix is a very small fix, it has not yet been deployed due to its significance and the need for extended testing. However, our results show the importance of the fix for mobile performance. We note that the bug fix also covers some cases that are addressed by the RTT-update patch, which means that a deployment of the bug fix would have been even more important than the algorithmic change. Nevertheless, the algorithmic change is generally valid and beneficial for scenarios with high delay variability.

We note that there are still a few strong outliers on either side that are not covered by either of the two patches. They are also a result of HyStart misinterpreting RTT variations caused by cellular uplink scheduling. While fixing them requires a slightly larger change, e.g., adding a simple filter to remove outliers, these are common effects of mobile network that are often not sufficiently considered when designing and evaluating transport and congestion control mechanisms, especially delay-based approaches. Ensuring that the variance of the base delay is handled appropriately is a very important aspect for performance in mobile networks.

V. Conclusion

In this paper we made a deep dive into the early Slow Start exit problem of TCP CUBIC when combined with HyStart and used in cellular networks. We explicitly tracked back the problem to the delay increase detection mechanism of HyStart, which misinterprets RTT drops as network congestion. Showing measurements of per-packet RTTs variations, which are inherent in cellular networks because of uplink scheduling principles, we have outlined in detail that these RTT variations, and the false interpretation of them, are the root cause for the early Slow Start exit issue.

In this paper we focused on a controlled, testbed-based scenario without cross traffic or mobility. The setup allowed us to analyze the effect of early Slow Start exits in detail, and show the impact of this issue particularly for short flows, which is one of the dominant communication patterns for web traffic in mobile networks and significantly impacts, e.g., H2 performance [16]. Although there are other factors like network load or signal strength having significant impact on performance [30], in this paper we aimed to separate such effects and address them independently. Nonetheless, the same issue can be observed in the real world and is particularly noticeable for short flows (cf. Sec. III).

Beyond that, we have shown how two straightforward HyStart changes improve the performance of TCP CUBIC significantly. In our LTE testbed the mean throughput of a $5\,\mathrm{MB}$ data transfer increases by 10.5 %. The performance gain is solely due to a larger CWND size at the Slow Start exit point, which is set by HyStart. Especially, as end-to-end delay is more and more decreasing, e.g. due to edge computing deployments and content delivery networks, where the nearest data center (i.e. server) can often be reached with only $10-20\,\mathrm{ms}$, this problem becomes prevalent and cannot be ignored. We further argue that with more appropriate testing under mobile conditions, it would have been possible to avoid such issues in the HyStart algorithm and implementation from the start. A stronger focus on cellular networks should be given when designing protocols and evaluating their performance, in particular congestion control algorithms.

In the future we would like to further look into the small set of remaining, more problematic cases of premature Slow Start exit. We anticipate that these cases can be addressed by methods to filter outliers when finding the minimum RTT, which could be used to enhance HyStart, but other transport mechanisms as well. Moreover, there is room for improvements for entirely new Slow Start algorithms that are better suited for cellular networks, where the base RTT is generally not constant.

We again underline the importance of considering mobile network characteristics when designing protocols and evaluating their performance given the steady increase in importance of mobile connectivity for the web.

References

[1] K. R. Fall and W. R. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*. Addison-Wesley, 2011.

[2] J. Iyengar and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport," Internet-Draft draft-ietf-quic-transport-33, Dec. 2020. Work in Progress.

[3] P. Yang, J. Shao, W. Luo, L. Xu, J. Deogun, and Y. Lu, "Tcp congestion avoidance algorithm identification," *IEEE/ACM Transactions on Networking*, vol. 22, no. 4, pp. 1311–1324, 2014.

[4] D. Havey, "Windows transport converges on two congestion providers: Cubic and ledbat." TECHNET Microsoft Network Blog, November 2018.

[5] S. Ha and I. Rhee, "Taming the elephants: New tcp slow start," *Computer Networks*, vol. 55, no. 9, pp. 2092–2110, 2011.

[6] R. Robert, E. Atxutegi, A. Arvidsson, F. Liberal, A. Brunstrom, and K. Grinnemo, "Behaviour of common tcp variants over lte," in *2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, 2016.

[7] Statista, "Percentage of mobile device website traffic worldwide from 1st quarter 2015 to 3rd quarter 2020." URL: https://www.statista.com/statistics/277125/share-of-website-traffic-coming-from-mobile-devices/, Oct. 2020. Retrieved: 2020-01-04.

[8] W3Techs, "Usage statistics of operating systems for websites." URL: https://w3techs.com/technologies/overview/operating_system, Oct. 2020. Retrieved: 2020-10-19.

[9] HTTP Archive, "State of the web." URL: https://httparchive.org/reports/state-of-the-web, Oct. 2020. Retrieved: 2020-12-30.

[10] J. Garcia, S. Alfredsson, and A. Brunstrom, "Delay metrics and delay characteristics: A study of four swedish hsdpa+ and lte networks," in *2015 European Conference on Networks and Communications (EuCNC)*, pp. 234–238, June 2015.

[11] A. Gurtov, *Effect of Delays on TCP Performance*, pp. 87–105. Boston, MA: Springer US, 2002.

[12] M. Yavuz and F. Khafizov, "Tcp over wireless links with variable bandwidth," in *Proceedings IEEE 56th Vehicular Technology Conference*, vol. 3, pp. 1322–1327 vol.3, Sep. 2002.

[13] M. Scharf, M. Necker, and B. Gloss, "The sensitivity of tcp to sudden delay variations in mobile networks," in *Networking 2004* (N. Mitrou, K. Kontovasilis, G. N. Rouskas, I. Iliadis, and L. Merakos, eds.), (Berlin, Heidelberg), pp. 76–87, Springer Berlin Heidelberg, 2004.

[14] J. Huang, F. Qian, Y. Guo, Y. Zhou, Q. Xu, Z. M. Mao, S. Sen, and O. Spatscheck, "An in-depth study of lte: Effect of network protocol and application behavior on performance," *SIGCOMM Comput. Commun. Rev.*, vol. 43, pp. 363–374, Aug. 2013.

[15] E. Atxutegi, Å. Arvidsson, F. Liberal, K.-J. Grinnemo, and A. Brunstrom, *TCP Performance over Current Cellular Access: A Comprehensive Analysis*, pp. 371–400. Cham: Springer International Publishing, 2018.

[16] N. Agarwal, M. Varvello, A. Aucinas, F. Bustamante, and R. Netravali, *Mind the Delay: The Adverse Effects of Delay-Based TCP on HTTP*, pp. 364–370. New York, NY, USA: Association for Computing Machinery, 2020.

[17] E. Dahlman, S. Parkvall, and J. Sköld, "Chapter 13 - power control, scheduling, and interference handling," in *4G LTE/LTE-Advanced for Mobile Broadband* (E. Dahlman, S. Parkvall, and J. Sköld, eds.), pp. 265 – 299, Oxford: Academic Press, 2011.

[18] S. Ha and S. Hemminger, "Tcp cubic in the linux kernel." URL: https://github.com/torvalds/linux/blob/master/net/ipv4/tcp_cubic.c, Jan. 2021. Retrieved: 2020-01-06.

[19] I. Rhee, L. Xu, S. Ha, A. Zimmermann, L. Eggert, and R. Scheffenegger, *CUBIC for Fast Long-Distance Networks*, 2018.

[20] T. Koyama and K. Aoki, "Slow start algorithm for mobile broadband networks including delay unrelated to network congestion," in *2015 International Conference on Computing, Networking and Communications (ICNC)*, pp. 148–152, 2015.

[21] P. Balasubramanian, Y. Huang, and M. Olson, "Hystart++: Modified slow start for tcp - draft-ietf-tcpm-hystartplusplus-01." IETF Internet-Draft, January 2021.

[22] S. Floyd, "Limited slow-start for tcp with large congestion windows." IETF RFC, March 2004.

[23] X. Zhang, *LTE Optimization Engineering Handbook*. John Wiley & Sons, 2018.

[24] NRG Systems GmbH, "Nrg systems ipu675 system." URL: https://www.ipu-system.de/produkte/ipu675.html, Sept. 2020. Retrieved: 2020-09-25.

[25] Sierra Wireless S.A., "Airprime mc7430 m2m module." URL: https://www.sierrawireless.com/products-and-solutions/embedded-solutions/products/mc7430/, Sept. 2020. Retrieved: 2020-09-25.

[26] J. Dugan, S. Elliott, B. A. Mah, J. Poskanzer, and K. Prabhu, "iperf - the ultimate speed test tool for tcp, udp and sctp." URL: https://iperf.fr/, 2020. Retrieved: 2020-07-15.

[27] E. W. Biederman and N. Dichtel, "ip-netns - process network namespace management." URL: https://man7.org/linux/man-pages/man8/ip-netns.8.html, June 2020. Retrieved: 2020-07-14.

[28] M. Kerrisk, "Namespaces - overview of linux namespaces." URL: https://man7.org/linux/man-pages/man7/namespaces.7.html, June 2020. Retrieved: 2020-07-14.

[29] F. Baumgarten, "netstat - print network connections, routing tables, interface statistics, masquerade connections, and multicast memberships." URL: https://man7.org/linux/man-pages/man8/netstat.8.html, July 2020. Retrieved: 2020-08-05.

[30] A. Nikravesh, D. R. Choffnes, E. Katz-Bassett, Z. M. Mao, and M. Welsh, "Mobile network performance from user devices: A longitudinal, multidimensional analysis," in *Passive and Active Measurement* (M. Faloutsos and A. Kuzmanovic, eds.), (Cham), pp. 12–22, Springer International Publishing, 2014.