CDBE: A cooperative way to improve end-to-end congestion control in mobile network

Zhenzhe Zhong*, Isabelle Hamchaoui*, Alexandre.Ferrieux*, Rida Khatoun†, Ahmed Serhrouchni†
* Orange Labs, Lannion, France
{zhenzhe.zhong, isabelle.hamchaoui,alexandre.ferrieux}@orange.com
† Télécom ParisTech,Paris, France
{rida.khatoun,ahmed.serhrouchni}@telecom-paristech.fr

***Abstract*—The advancing MAC/PHY technique and architecture in the mobile network allows the DownLink (DL) capacity in radio access network (RAN) to vary from Kilo-Byte per-second level up to Giga-byte per-second level in a glimpse. The existing TCP congestion control algorithms (CCA) were not designed for such dramatic variability. In this paper, we proposed an improvement for end-to-end congestion control, called Client Driven Bandwidth Estimation (CDBE), which allows TCP clients to cooperate with its CDBE. The cooperation can enhance the down-stream (DS) performance of the connections, even in the mobile network. The CDBE client can measure the available bandwidth (BW) on the bottleneck and inform the server of the BW estimation (BWE) result. Unlike existing mobile cross-layer congestion control proposals, the proposed algorithm on TCP client does not directly invoke information from UE MAC/PHY layer module but implicitly reflect the varying cellular BW when the mobile last hop is the bottleneck of the network. The CDBE TCP server uses the received BWE value to calculate its pacing rate and congestion window, and it further calibrates the result according to the variation of down-stream delay (DSDL). The state transition in the server can react rapidly to eNB BW and queue variation. Sets of NS3 system simulation in a LTE network is conducted with BBR and CQIC as the baseline. The result shows that the proposed algorithm can improve the end-to-end throughput and good-put while keeping the DS delay at a low level.**

I. Introduction

Transport Control Protocol (TCP) is the typical congestion control choice to discipline the end-to-end (E2E) traffic. In legacy TCP, congestion window ($cwnd$) and slow start threshold ($ssthresh$) are the two main features which govern the TCP transmission pattern. According to the selected congestion control algorithm (CCA), the update of $cwnd$ and $ssthresh$ are confined by a reaction to each Acknowledgement (ACK) in SS and CA or by a Re-transmission Timeout (RTO). Some existing TCP CCA, e.g. Westwood [1], Vegas [2], and most recent BBR [3], may roughly probe the network capacity and use it as the aid of congestion control, and note that BBR is the first widely deployed congestion-based CCA without a strict additive increase multiplicative decrease (AIMD) behaviour. As a consequence, BBR gives TCP a better capacity to react to BW fluctuations than typical AIMD CCA [4].

The evolution of cellular networks raises a new challenge for BWE based CCA design. On the one hand, an appropriate choice of scheduling method and modulation and coding scheme (MCS) can reduce the possibility of radio transmission failure. This strategy achieves higher radio resource utilisation from media access (MAC) perspective of view. However, from per UE perspective of view, they should experience a fluctuation of instantaneous radio DL BW according to eNB scheduling result [5]. On the other hand, the lossy nature of the radio channel is handled by ARQ/HARQ in Radio Link Control (RLC) and MAC layers in the LTE network. Multiple levels of retransmission mechanism can introduce a certain amount of delay according to RAN configuration [5, 6]. Hence from the E2E point of view, a delay variation instead of the loss is observed. As a result, the loss detection based CCA may cause bufferbloat [7] and can not perform well in such scenario [8, 9, 10, 11]. Furthermore, the LTE uplink and downlink are asymmetric. The uplink uses the robust transmission strategy [12] and power saving configuration [13] which can be a constraint of the data communication.

Centred around these challenges, a TCP client driven BWE (CDBE) method for the fast BW variation is proposed to address the problems. In this CCA architecture, the client and server of the TCP connection can cooperate with each other. A CDBE client can engage in end-to-end CCA decision loop by offering BWE feedback. When the bottleneck is working on the optimum point as described in [14], the feedback is the BW from bottleneck. The benefit of doing so is that the non-application limited DS traffic can be delivered without considering the upstream bottleneck. When the last hop RAN is the bottleneck of the network, a fairly accurate last-hop BWE is achieved by dual-window BWE method. While the bottleneck is elsewhere in DS, the BWE method can also reflect the core network bottleneck BW. The measurement result is further piggybacked to the server by introducing a new "BW" optional field on TCP header in our experiment. The server then exploits the estimation result to calculate traffic shaping elements, $cwnd$ and pacing interval (PI). Different gains are applied to computed elements, according to the DS delay, to avoid bufferbloat or to push more data into the network if there are available BW in the bottleneck.

The rest of the paper is organised as follows: Section II gives a brief overview of E2E CCA and related benchmarks. Section III specifies the detail of dual-window method of BWE in CDBE client. Section IV describes the detail of state transition of on server side. Finally, we conduct a performance investigation of the proposed CCA with up to date benchmark in Section V. Last but not least, we conclude the paper in section VI.

II. Overview and related work

*1) **Conventional congestion control algorithms***

The E2E congestion control in a network is broadly studied as variants of TCP CCAs. From the famous early variant Tahoe [15], Reno [16] and new Reno [17], to the most popular Cubic [18] in current operating systems, TCP CCAs employ loss-based and the AIMD-window based control manner. The design was not aiming to efficiently achieve a high BW utilisation in a BW varying network. TCP Vegas is one of the earliest delay-based congestion control algorithm. This pure delay-based CCA can be too polite when competing with loss-based CCA [19].

*2) **Recent congestion control proposals***

As a successor of Vegas, BBR improves the delay-based concept to congestion-based by taking both BWE and bottleneck queue management into consideration. It remarkably improves the E2E performance compared to conventional CCA variances. Verus [20] uses Round Trip Time (RTT) gradient to infer the congestion degree in the network and evolve the current sending window size from the previous window size. The google congestion control (GCC) [21] is the main congestion control for WebRTC. It evolves the BW report on the client side by a factor. The server uses the report to update the transmission pace for the current video frame quality.

*3) **Congestion control with explicit cross-layer information***

Existing efforts have tried to improve the E2E BW utilisation by invoking the cross-layer information directly, and they assume that the RAN is the bottleneck of the network: *piStream* [22], invoke lower layer information (MAC, PHY) to improve the E2E performance and get rid of the limit of $cwnd$ linear growth. The difficulty of deploying this method is that they need the support of designated hardware. Channel Quality Indicator Control (CQIC) is a cross-layer CCA in 3G [23] and 4G network [24]. It is designed to address the inaccuracy of TCP BWE and enhance the BW utilisation in a mobile network. This CCA algorithm also needs the lower layer hardware support to invoke CQI or DCI [24]. The translated MAC layer throughput will be sent to the server for pacing rate and cwnd calculation. The mobile throughput guidance [25] is proposed to locate the available BW in RAN from eNB perspective of view. However, the last hop in the RAN may not always be the bottleneck. The other common drawback of the specific cross-layer information is that the hardware changes are required on either endpoint of the connection or the middle point on the path. Last but not least, the direct use of throughput from a specific layer (e.g. MAC) may overestimate the bottleneck since the higher layer retransmission mechanism, e.g. RLC ARQ is not considered [26].

These works inspire the idea of CDBE Server-Client cooperative CCA. As illustrated in Fig. 1, the client report manner can avoid taking upstream bottleneck and RAN lower layer overhead into BWE. The bottleneck elsewhere is also taken care of when necessary. The challenge is to implicitly estimate BW without directly invoking the lower layer information while achieving fairly high accuracy. Moreover, the CDBE server should also be able to infer the status of eNB buffer: when to probe for higher available BW and when to make housekeeping move to ease queue pressure. The design will be introduced in following two sections.

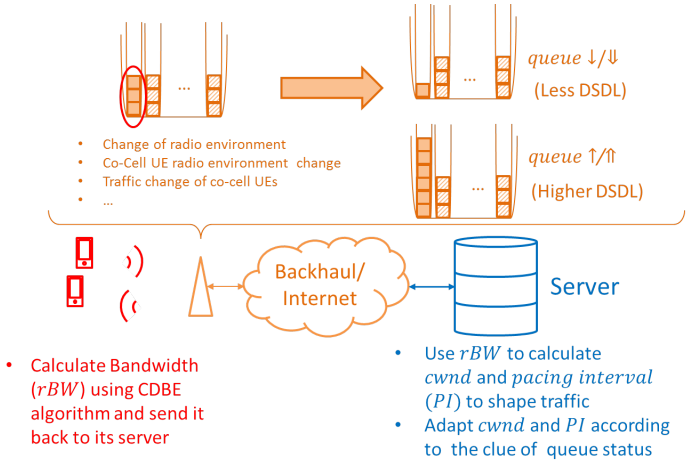


Fig. 1: Elements of CDBE congestion control

III. Bandwidth estimation on CDBE Client

Different from the existing server-side BWE methods listed in [27], the algorithm runs on the UE side, and it is a continuous measurement. The CDBE BWE aims to follow the BW variation of the mobile network in ms level. Hence in this work, all the TCP clients run in UEs, and the following UEs also represent CDBE clients.

A. Bandwidth Estimation method in UE

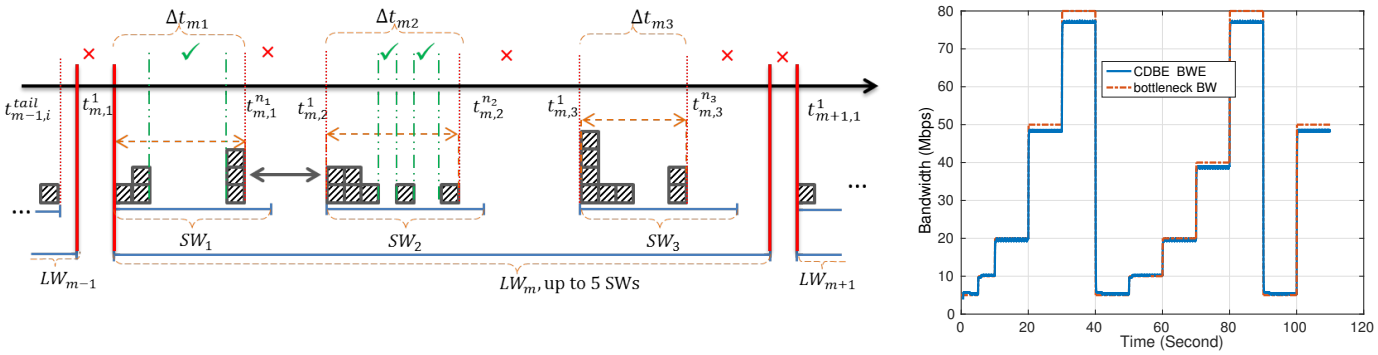
1) Principle of design

The arriving rate can be simply abstracted as follows:

$$R(t) = \frac{\sum Pkts}{Duration} \tag{1}$$

The duration can be calculated from the system timer or timestamp value. Simply use a hard-coded constant duration to calculate arriving rate can cause the following issues: 1. involve an unnecessary volume of time to cause underestimation, since we do not know how much spare time is involved in the head and tail of the window, or, 2. over-estimate due to the bursty arrival caused by PDCP in order delivery mechanism. When this constant value is set too large, the underestimation in the former point is more significant, and when the window is too small, the latter overestimation is more obvious. Hence, the choice of constant window as duration may cause in accurate rate calculation on the client side like that in GCC, CQIC, piStream, etc.

The principle of the design is to filter out the unnecessary time gap in the BW calculation but not to eliminate gap caused by the delay. To address the issue, a two-window based BWE method is proposed for this very first version of CDBE: The CDBE client is responsible for measuring the BW samples (sBW) of each sample windows (SW) and use their average to update the report (rBW) in every long window(LW) epoch. The SW captures the total amount of received data segments to calculate sBW, whereas the latter epoch considers all the



(a) (b)

Fig. 2: (a) Illustration of CDBE filter function on UE (b) Validate CDBE BWE method in a wired network with UDP traffic

included sBW to calculate rBW. Once the rBW is updated, the value is converted to 32 bit BW option in ACK. The time limit of SW is W_s, and for LW it is denoted as W_l. The size of W_s and W_l can affect the accuracy of estimation. The W_l is five times W_s period in current configuration. This allows no more than 5 sBW in each LW. A LW begins with the head of its first SW and ends with the termination of its last SW. To guarantee the timeliness of the rBW, the W_l is set to the same as RTT since it takes at least one RTT to reflect the current rBW back to the client.

The implementation of CDBE client requires to turn on the time-stamp option for a better RTT accuracy. The redesign of TCP TS field may be required [28]: More than five different granularity exists in current TCP implementation. In the evaluation of this work, the TS granularity is synchronised to 1ms.

2) Detail of BWE method

Fig. 2(a) is used to aid the understanding of the proposed BWE method. A cube in the Fig. 2(a) represents an arriving TCP data segment. Each LW epoch begins with first arriving TCP data segment of its first sample window SW_1 at t_{11}. Upon the arrival of each data segment, a CDBE client calculates the time difference (Δt) between now and the head of current SW_i. Once the received data segment falls outside of SW_i, it becomes the head of the next SW_{i+1}. A BW sample sBW_i is calculated using the equation below:

$$sBW_i = \frac{\sum_1^n P_n}{time_i} \tag{2}$$

where n is the total number of segments received in SW_i, and P_n is the size of the n^{th} packet in SW_i. $time_i$ is selected from one of the three possible value: minimal duration (Δt_{min}), W_s and explicit time difference between the first and the last segment in SW ($\Delta t_i = t_{in} - t_{i1}$):

$$time_i = \begin{cases} W_s & \text{if } \Delta TS_i >= 2 * max(\Delta t_i, \Delta t_{min}) \\ \Delta t_{min} & \text{if } \Delta t_i == 0 \\ \Delta t_i & Otherwise \end{cases} \tag{3}$$

TCP timestamp is used to support the decision making. When the timestamp difference ($\Delta TS_i = TS_{in} - TS_{i1}$) of corresponding data segments is larger than Δt_i, it means there must be some level of delay in the network. Some part of the delay may comes from the HARQ/ARQ mechanism in LTE which guarantees the in order delivery of its data packet according to our observation: Once a burst of segment reception happens, there is a possibility that the delivery of the first $(j-1)^{th}$ segments is obstructed by j^{th} segment due to several rounds of unsuccessful retransmission of $(j-1)^{th}$. This phenomenon is irrelevant to the BW reduction, and it can be averaged out by taking mean of samples in a LW. We also propose that when ΔTS_i is more than twice of $max(\Delta t_i, \Delta t_{min})$, we should use W_s instead of Δt to avoid potential over-estimation in current SW_i. This approach tackles most of the over-shooting problem and performs well in our simulation. However, in some extreme cases, if the RLC/PDCP caused bursty reception of data is severe, the Δt_i can be very small. As a result, a large number of packets can arrive with $\Delta t = 0$. A minimum time duration t_{min} is introduced to make sure the valid calculation in equation 2. In our current implementation, t_{min} is a constant value of 2ms.

Once the current segment falls outside the current LW epoch, the UE will calculate the average of the BW samples in LW. Further, the valid latest rBW is calculated as the smooth averaged with the prior report value (rBW_{m-1}):

$$rBW_m = \beta * rBW_{m-1} + (1-\beta) * \overline{rBW}_m \tag{4}$$

where,

$$\beta = \begin{cases} \dfrac{0.4}{0.4 + (t_m^{first} - t_{m-1}^{last})} & \text{if } t_m^{first} - t_{m-1}^{last} > W_S \\ 0.4 & Otherwise \end{cases} \tag{5}$$

This low pass filter manner reports a smoother rBW and mitigate the sharp variation of $cwnd$ and PI when RTT is small. If rBW_{m-1} record is too old, it should not affect the current result too much, hence if time difference of two LW is more substantial than W_s, β is formulated as an inverse proportional to the time gap between the two LWs. Last but not least, under the current configuration, the initial BW report is 1 MBps.

B. Validate CDBE BWE method with saturated traffic

To validate the proposed method, we first test the CDBE client BWE algorithm in a fixed network. We simulated a two-hop wired network with one varying bottleneck and minimum round trip delay of 80ms. The bottleneck BW changes in the following sequence: 5, 10, 20, 50, 80, 5, 10, 20, 40, 80, 5, 50 Mbps, at 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 second respectively. The second hop has 10Gbps BW which guarantees the bottleneck limits the traffic. The host nodes make UDP download to saturate the bottleneck which guarantees the BWE in the client can reflect the maximum BW in the bottleneck. The

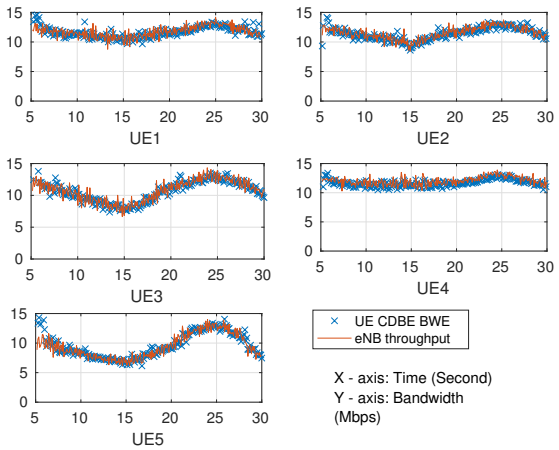


Fig. 3: Validate CDBE BWE method in LTE with UDP traffic

result is shown in Fig. 2(b): regardless of lower layer overhead, the BWE tracks the discrete downstream BW variation well.

We further simulate the CDBE BWE in five mobile UEs under UDP-download scenario for 30 seconds. Five co-cell UEs locate on 50, 100, 120, 150, 180 metres away from eNB. The download starts from 5^{th} second and the UEs move away from eNB for first 10 seconds at 6m/s. After that, the UEs then move toward eNB at the velocity of 10 m/s for next 10 seconds. In the last five second of simulation, the UEs leave eNB again, but their speed is 20m/s. As illustrated in Fig. 3, the CDBE algorithm in UEs can keep track of continuous BW variation. Hence we confirm that the proposed algorithm is capable of following the track of saturated traffic.

IV. CDBE STATE TRANSITION MODULE IN SERVER

In non-application limited mode, the CDBE server uses received rBW to calculate the $cwnd$ and PI:

$$PI = \frac{pktSize}{G_{pacing} * rBW} \tag{6}$$

$$cwnd = G_{cwnd} * rBW * RTT_{min} \tag{7}$$

Pacing interval is the main character to shape the traffic, while $cwnd$ is the minor congestion control factor to guarantee that the amount of traffic does not exceed scaled BW delay product according to current state. According to the analysis in Section III-B, it is necessary for the server to send enough backpressure for the bottleneck to reach the optimum working point for the BWE method to track the BW variation accurately. In the meantime, the queue should also be maintained on a relatively low level. The server uses the continuous probing method to feed the bottleneck with enough data while also take care of the queue in the bottleneck buffer. We will discuss transition logic and the definition of the states in the following two subsections.

A. State definition and impact of parameters

The CDBE server considers the downstream delay (DSDL) variation as the main indicator of queue state in eNB. Hence in our current implementation, the factor to optimise is DSDL, which is directly available from TCP TS option. The DSDL growth caused by the server's own traffic relates to an eNB DL

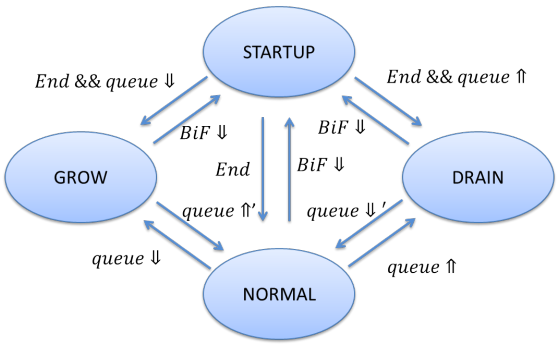


Fig. 4: CDBE server state transition

queue build up while DSDL drop corresponds to a draining eNB DL buffer. The server should *increase*, *decrease* and *maintain* the calculated $cwnd$ and PI due to the queue size variation. In such case, the states defined for CDBE are: *STARTUP*, *NORMAL*, *DRAIN*, *GROW* and their transition is illustrated in Fig. 4. To control the amount of traffic, pairs G_{pacing} and G_{cwnd} are designed for each states. These value of the pairs are inherited from BBR: $2/ln2, 1, 0.75, 1.25$ are the G_{pacing} for *STARTUP*, *NORMAL*, *DRAIN*, *GROW* respectively and the G_{cwnd} for corresponding states are $2/ln2, 2, 2, 2$.

Sender travels among the states in a time-windowed (TW) manner ($W_{update} = RTT_{min}$) based on the sign of DSDL variation. The detail of each state is given below:

■ ***STARTUP***: This is the first stage a sender/server enters. The server assumes that there is no pipe in the eNB or not enough data in the flight for the current connection. Hence CDBE sender uses more aggressive G_{pacing} and G_{cwnd} to build up the pipe in the eNB as fast as possible. This approach squeezes more and more of the available BW for m round trip time. Once the startup-count is larger than m or the growth of received BW report is less than 10% for two continuous minimum RTT. Similar to BBR startup, this is a dynamical variant of slow-start and keeps similar TCP SS performance for short flow. In *STARTUP*, the sender has begun keeping on monitoring the DSDL and BWE for an earlier preparation for state transition among the other state.

■ ***GROW***: The main purpose of the *GROW* state is to probe the available BW in a BW varying network. For example in LTE cellular, if the radio and co-cell traffic condition are good enough, we should observe no queue up when we use the original rBW from UE to shape traffic. After n period of stable DSDL, the server will enter *GROW* to send at a gain of 1.25. The UE will observe the higher amount of received data segments if there are some available spare BW and the corresponding rBW will increase exponentially. Before the sign of $queue\Uparrow'$ is met, server will stay in *GROW*. This manner allows faster progress of BW increment compared to BBR. The conditions marked in Fig. 4 will be explained in next subsection.

■ ***DRAIN***: *DRAIN* reduces the bottleneck queue. $queue\Uparrow$ corresponds to the inflating buffer caused by the server's own traffic. Once the condition of DRAIN is met, the server will try to ease the potential bufferbloat caused. Under the current configuration, the server will keep draining the queue until

queue$\Downarrow'$ is shown. The server will use *NORMAL* for one TW to wait and see the dropping trend of DSDL. If DSDL does not fall back to $DSDL_{min}$, a new $DSDL_{min}$ may be updated. This approach stables the DSDL at a reasonable level, and the server will go to *NORMAL* to maintain the amount of data.

■ ***NORMAL***: *NORMAL* is used to avoid the oscillation between *GROW* and *DRAIN*. It allows the server to observe that how the network reacts to its gain pairs variation. In *NORMAL* state, the server will observe the DSDL for a couple of TW. As long as the trend of DSDL in last TW does not vary continuously on the same trend, sender assumes the buffer in the bottleneck buffer is about stable, and sender stays in *NORMAL*.

B. State transition condition in CDBE server

The server maintains the record of DSDL and the received rBW to appraise the result of the queueing inspection. The DSDL trend, the change of rBW, and current state decide the next state. The DSDL can be obtained by calculating the difference between TSecr and TSval of TCP timestamp option for each received segment. Due to the bursty scheduling manner in LTE network, the packet arriving delay may vary widely [20]. To observe the steady trend of DSDL variation in the last RTT, the sender should check the average of the DSDL sample recorded in each recorded each TW :

$$A_{DSDL_i} = \sum_{j=1}^{T_i} \frac{DSDL_j}{T_i} \tag{8}$$

The average distance of DSDL samples from minimum $DSDL_{min}$ is also considered to eliminate the DSDL variation:

$$\Delta DSDL_i = \sum_{j=1}^{T_i} (DSDL_j - A_{DSDL_{i-1}}) \tag{9}$$

Further, minimum DSDL ($DSDL_{min}$) and *BDP* are also use as a clue. The sender decides its state in the next sample period according to these evidence shown below:

☐ **Obvious Queue growth (*queue*↑)**:

$A_{DSDL_i} > (1+\alpha) * A_{DSDL_{i-1}}$ and $\Delta DSDL_i > 0$,
or $A_{DSDL_i} > (1+\beta) * DSDL_{min}$

When the DSDL shows the trend of growth, it means the queue is possibly building up. When the average DSDL is larger than $(1+\alpha) * DSDL_{min}$, it will also be considered as one potential hint for *DRAIN*. Concerning the queue up sign may be the result of the traffic from other traffic sources, α is set to 0.1 to tolerate the queuing pressure from other traffic sources. It not only guilds the queue length built by the server's own traffic but also prevents the server from draining the queue frequently as Vegas when facing cross traffic.

☐ **Obvious Queue decrease(*queue*↓)**:

$A_{DSDL_i} < (1-\alpha) * A_{DSDL_{i-1}}$ and $\Delta DSDL_i < 0$,
or $A_{DSDL_i} < (1+\alpha) * DSDL_{min}$

When the DSDL shows the trend of decrease, it means the queue is possibly draining, or the route has changed. When the average DSDL is lower than $(1+\alpha) * DSDL_{min}$, it will also be counted as one hint for potential *GROW*: we can probe for more available BW. We record this trend to confirm the further firm Queue decrease.

☐ **Firm Queue growth(*queue*⇑)**:

1. $A_{DSDL_i} >= (1+\beta) * A_{DSDL_{i-1}}$, or
2. n continuous (queue↑),

Any of the two conditions above is met, server will go to *DRAIN* . The reasons of setting $n = 2$ are: 1.after two continuous queue↑, the DSDL growth is more than 21% which is close to $\beta = 0.25$ in the first condition above, and 2. average DSDL continuously less than minimum DSDL record for two RTT periods. $\beta = 0.25$ is also conjugated with the 25% gain value used in ***GROW***.

☐ **Firm Queue decline or Probe (*queue*⇓)**:

1. $A_{DSDL_i} < (1-\beta) * A_{DSDL_{i-1}}$, or
2. m continuous (queue↓)

Any of one the two conditions above is met, server will go to *DRAIN* to use more aggressive G_{pacing} to capture the potential BW growth in bottleneck.

☐ ***GROW* caused queue growth(*queue*⇑′)**:

queue↑ or $A_{DSDL_i} >= (1+\beta) * A_{DSDL_{Gstart}}$

Server quit *GROW* after it observes an certain amount of DSDL growth (compared to the $A_{DSDL_{Gstart}}$ at the beginning of current *GROW* period).

☐ ***DRAIN* caused queue decline (*queue*⇓′)**:

queue↓ or $A_{DSDL_i} <= (1-\beta) * A_{DSDL_{Dstart}}$

Server quit *Drain* after it observes reduction of a certain amount of DSDL reduction (compared to the $A_{DSDL_{Dstart}}$ at the beginning of current *DRAIN* period).

☐ **Bytes in flight lower than expectation (*BiF*⇓)**:

BiF $< \gamma * BDP$

When the amount of bytes in flight is lower than expectation,e.g. after a application limited transmission period, the server can Choose to enter *STARTUP* again with a reserved $StartupCount$ value.

☐ **To exit *STARTUP* (*END*)**:

Server exit *STARTUP* when counter reaches $StartupCount$ or the data increase is less than α for two continuous minimum RTT.

For the rest of the situation, the server will consider the queue state in the bottleneck as stable and stay in ***NORMAL*** for most of the time. One more benefit of using this module is that it can work with any BWE algorithm (on TCP client or Server). It is proved later in the simulation section.

V. Simulation and result discussion

A. Simulation configuration

An NS3 simulation is conducted to measure the performance of CDBE server and the CDBE architecture as a whole. In the simulation, a remote server is the sender and all 5 UEs are downloading from the server throughout the simulation. The size of the download is set as 10MB and 1MB to test the performance of traffic amount. The remote server is connected to P-gate way directly to simulate the CDN download scenario. The connection starts from 3rd second, and each full simulation lasts 15 seconds. The end-to-end performance of CDBE is evaluated against CQIC, CQIC client+CDBE server

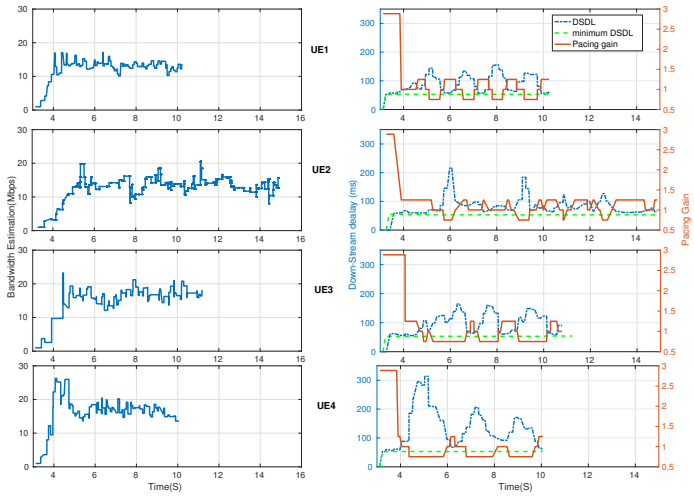


Fig. 5: Bandwidth estimation (left), Gain, DSDL and $DSDL_{min}$ in one experiment (right)

(CQIC-S), and TCP BBR. We have shown that CQIC can outperform Cubic and Westwood in this scenario, hence BBR is a reasonable TCP baseline compared to legacy TCP CCA. Note that the performance of CQIC-S is also used to validate the assumption that CDBE server can cooperate with any TCP client with BWE report capability and improve their performance by taking care of the Queue states in the bottleneck. All the TCP clients in simulation enable delay ack with the configuration of DelAckCount=2, DelAckTimer=40ms. LTE

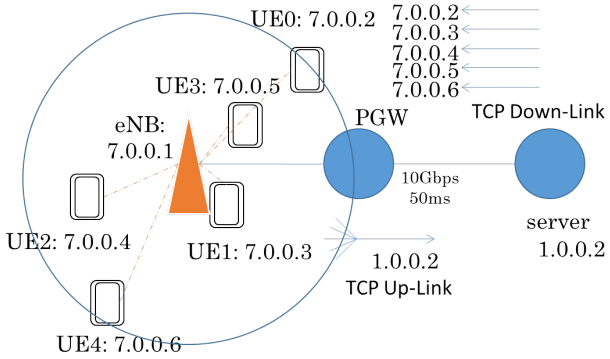


Fig. 6: Simulation topology and concept illustration

module and Internet module have been customised to carry out RLC Acknowledged Mode (AM) re-segmentation, RLC drop tail buffer and CDBE implementations. We consider a typical outdoor scenario that all the UEs are attached to one single eNB locating on the centre of a disc with a radius of 400m. 5 UEs are moving at random walking speed of 4m/s, and 250 different drops of an arbitrary initial position are tested for the statistic. The detail of the configuration is shown in Table I and the topology is shown in Fig.6.

B. Performance of CDBE server

The main simulation result is listed in Table II. To validate the performance of CDBE server, we use CQIC enabled UE to

Topology	Number of UEs	5
	Number of eNBs	1
	Mobility model	Random walk
	UE Velocity interval (m/s)	[1, 5]
LTE RAN	Path-Loss Model	Cost231
	Fading model	EPA
	Tx Power (dBm)	46 (eNB),24 (UEs)
	Noise Figure	5 (eNB),9 (UEs)
	Scheduler type	Proportional Fair
	Number of Resource Block	100 (DL/UL)
	RLC configuration	AM
	RLC buffer size	(1Mb per flow)
	DL/UL frequency	2120 / 1930 MHz
Core Network	Delay (ms/hop)	50
	Capacity (GBps)	10

TABLE I: Simulation Configuration

	BBR	*CQIC*	*CQIC-S*	CDBE
End-to-end Downstream(1MB and *10MB*)				
DS TP (Mbps)	22.73 37.01	35.82 58.98	37.01 58.88	32.88 56.3
DSDL (ms)	66.45 115.87	127.1221 153.93	134.22 144.9	90.17 99.91
End-to-end overall				
Goodput (Mbps)	3.1427 6.47	5.08 7.50	5.32 7.56	4.69 7.26
RTT(ms)	0.6235 0.7708	1.0630 1.1741	1.0425 1.0366	0.7399 0.9130
Fairness (Goodput)	0.7907 0.6135	0.8164 0.6360	0.8250 0.6445	0.8093 0.6332

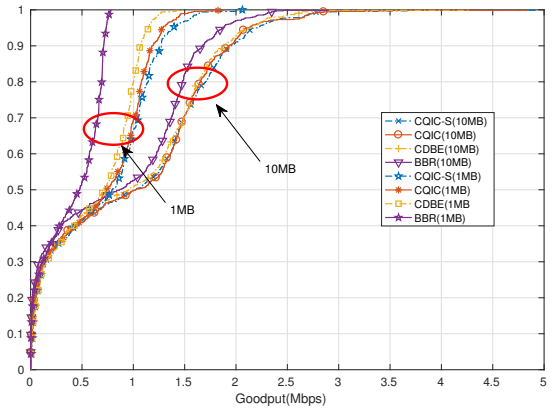
TABLE II: Compare the CDBE with Baselines

report the BWE. It is known that CQIC reports an aggressive last hop BWE as evaluated in [26]. When CQIC is armed with state transition enabled server, the overall performance improves in both 1MB and 10MB cases. This result shows that enabling CDBE state machine can be beneficial to a UE which is capable of making BW report. The exceptions are CQIC-S has higher DSDL in the 1MB case, and the DS throughput of CQIC-S is lower in 10MB case. The former is due to the CQIC implementation has more aggressive initial BW report than CDBE client, plus the offensive gain used by CDBE server in *STARTUP*. While the latter is caused by the frequent use of *DRAIN* gain to suppress the CQIC BWE, which involves MAC overhead but ignores retransmission. The benefit is that the load of eNB is relieved and the amount of loss and RTO reduce in this case. Overall, the goodput and RTT of CQIC type of client feedback BWE mechanism is improved by CDBE server since it somehow addresses the overestimation problem located by [26]. These result shows that CDBE server is able to cooperate with the client with aggressive BWE report and avoid bufferbloat in some degree.

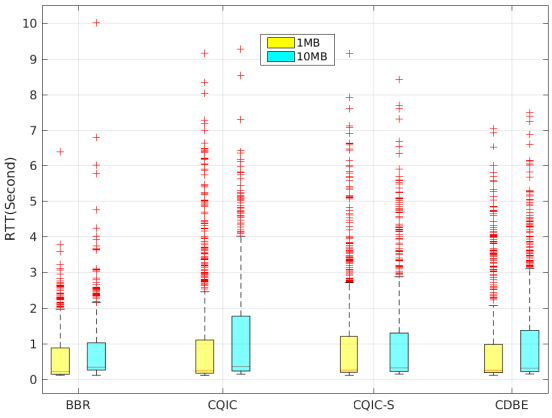
C. Per-flow analysis

Fig. 5 plots the tracks of one experiment in the data set. In this specific set of experiment, one of the UE is out of the connection range, and the distance of the rest of UEs are 46.55m,187.12m,143.57m and 109.13m away from eNB.

As we can see, the BW grows exponentially with *STARTUP* gains. Once *STARTUP* is over, the server will change its state according to the manner introduced in Section IV. When the DSDL is small enough (close to $DSDL_{min}$) for a continuous period, the server then tries to send more data to the network for UE to prob and grasp higher potential BW. When DSDL goes up, the server will go back to *NORMAL* to see whether this DSDL stays in a reasonable range once the DSDL is too high, the server should use reserved G_{pacing} to send so that the bottleneck can digest the queue. The result shows that this CCA architecture meets the objective of DS delay management.



(a)



(b)

Fig. 7: (a) Goodput CDF of CDBE, CQIC, CQIC-S and BBR and (b) RTT samples with 75% and 99% percentiles

D. System level result statistics, evaluation and discussion

The following subsections will discuss CDBE CCA architecture as a whole to compares to the benchmarks in four aspects:

1) ***Goodput*** *and* ***DS throughput***

Compared to BBR, CDBE has better average goodput. According to the CDF plot of the result in Fig. 7(a), the improvement in both 1MB and 10MB cases are not caused by the outliers. In 10MB case, most of BBR goes to *steady* and most of CDBE goes to its tour among *GROW*, *DRAIN*, and *NORMAL* states.

The performance of CDBE is close to CQIC-S regarding throughput and goodput, but there is still a gap. The gaps shown in Fig. 7(a) come from the fact that the BW report value of CQIC is more aggressive.

2) ***RTT*** *and* ***DSDL***

CDBE can outperform CQIC concerning DSDL and RTT. This result indicates that the higher BW value from CQIC BWE algorithm may not fit the queue well. Note that the configuration of parameters, e.g. α, β,etc., in sectionIV can affect the performance of CDBE. However, the DSDL is always better than CQIC or BBR in larger file case. This result shows that the state transition can minimise DSDL while keeping the DS throughput at a higher level.

BBR has a slower reaction since it will need four 8-minimum-RTT cycles to double to higher BW and need to wait for the 3-tier filter time out to decrease the BWE. The reactions, as a consequence, are on the second level. As for CDBE, as long as the server enters *GROW*, it can keep on growing as long as the hint of DSDL increase does not show. More importantly, the reported value can take effect immediately instead of using time 3-tiered filtered manner. This is the reason why CDBE has lower DSDL and higher DS throughput and higher goodput than BBR.

From the RTT point of view, CDBE has roughly 18% extra RTT for both 1MB and 10MB cases compared to BBR. It is mainly caused by a higher amount of ACK while the size of CDBE ACK is also significant. This extra amount of delay mainly comes from upstream, or more specifically in LTE uplink, due to the robust MCS choice [12]. One of the clues to support this point of view is that DSDL of CDBE is less than BBR in the 10MB case while it is the reverse in 1MB case. This result is because that the BW growth of CDBE happens one RTT faster than BBR. After one W_l the rBW can reflect the growth of BW, and after $W_l + ULDL$ (roughly a little bit more than one RTT) the grown BW becomes effective in CDBE server. As for BBR, the time for the first BW to take effect in the BBR startup period is 2RTT. Further, the in 1MB case, the connection stays mainly in startup for both BBR and CDBE and missing of compulsory drain period right after startup, CDBE can reveal higher DSDL and higher DS throughput. The RTT can be improved by reducing the frequency of BW report. Currently, we report rBW every ACK. This strategy results in an extra 32bits of traffic per ACK in Up-stream. It can be reduced to report in the period manner or upon every a few ACK.

3) ***Fairness***

Concerning the fairness of the network, the intra-protocol fairness of CDBE can be seen as an approximation to achieve equilibrium of proportion fair share of network capacity as described in [29]. When the bottleneck is in RAN, the UE side is capable of capture the variation of capacity and report the value to CDBE server. The server will reshape the traffic to fit the current capacity share among the UE and consider the potential bufferbloat. When the bottleneck is in somewhere else in the network, the capacity estimation of CDBE client is not limited by RAN BW share. Hence the BWE in the client is reflecting the BW of the actual bottleneck and try to approach the equilibrium point of actual congestion node. Thus, the client report manner can avoid the over-aggressive behaviour of TCP

BBR mentioned in [30].

For Inter-protocol fairness, we still need more experiment to discover test the friendliness of CDBE to legacy CCAs. The weakness of vegas has been taken into account and the second condition of *queue*↑ in section IV-B makes sure that the ***DRAIN*** only happens to take care of the bufferbloat caused by the server's own traffic. The CCA algorithm module in the server can be further altered to avoid too aggressive or too conservative performance in future research.

*4) **Deployment***

Last but not least, the deployment of CDBE needs the change from both the TCP server and TCP client, but only on the code in the transport layer. It simplifies the deployment compared to CQIC or other cross-layer design which requires the hardware changes. When compare CDBE to BBR, CDBE needs the support of the transport layer on the client side. Concerning the benefit mentioned earlier, we believe it worth deploying this interactive CCA architecture for connection endpoints to better utilise the future network capacity in both mobile and wired network for the future network.

VI. Conclusion and future work

In this paper, a novel congestion control architecture, named CDBE, is proposed to improve network end-to-end performance, especially in the mobile network. The two components of the proposed CCA architectures are bandwidth estimation algorithm in the client and a state transition module in the server. The CDBE client uses a dual-window algorithm to calculate the bandwidth and send it back to the server. The state machine in the server can decide how to use the reported value by judging the downstream delay extracted from TCP timestamp. As a result, as long as the variation of BW in the mobile network can be accurately tracked by the UE, the bufferbloat caused by the server's traffic can be suppressed by the server. The advantages of this CCA architecture are: 1.mitigate the influence of LTE uplink or bottleneck in up-stream path; 2. prompt estimation of the bandwidth and reserved housekeeping manner, 3. compatible CDBE server for different kinds of CDBE clients, 4. implicit bandwidth estimation algorithm on UE does not require lower layer hardware support, and 5. can also take care of the bottleneck "elsewhere".

The future work is to enhance the state machine and accuracy CDBE bandwidth estimation algorithm. The inter-protocol fairness is also being tested. A more concise way to make bandwidth report will also be investigated to replace the current per ACK report manner. Furthermore, the engineering detail for non-application limited traffic in CDBE server should also be specified during our future emulation. Last but not least, the influence of poor upstream bandwidth will also be investigated.

References

[1] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang, "Tcp westwood: Bandwidth estimation for enhanced transport over wireless links," in *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '01. New York, NY, USA: ACM, 2001.

[2] L. S. Brakmo and L. L. Peterson, "Tcp vegas: end to end congestion avoidance on a global internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 8, Oct 1995.

[3] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "Bbr: Congestion-based congestion control," *Queue*, vol. 14, no. 5, Oct. 2016.

[4] N. Cardwell, Y. Cheng, S. H. Yeganeh, and V. Jacobson, *BBR Congestion Control draft-cardwell-iccrg-bbr-congestion-control-00*, Google, Inc Std., July 2017.

[5] *Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures*, 3GPP Std.

[6] B. Levasseur, M. Claypool, and R. Kinicki, "Impact of acknowledgments on application performance in 4g lte networks," in *2015 International Conference on Computing, Networking and Communications (ICNC)*, Feb 2015.

[7] H. Jiang, Z. Liu, Y. Wang, K. Lee, and I. Rhee, "Understanding bufferbloat in cellular networks," in *Proceedings of the 2012 ACM SIGCOMM Workshop on Cellular Networks: Operations, Challenges, and Future Design*, ser. CellNet '12. New York, NY, USA: ACM, 2012.

[8] H. S. Park, J. Y. Lee, and B. C. Kim, "Tcp performance issues in lte networks," in *ICTC 2011*, Sept 2011.

[9] 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, no. 4, Aug. 2013.

[10] B. Nguyen, A. Banerjee, V. Gopalakrishnan, S. Kasera, S. Lee, A. Shaikh, and J. Van der Merwe, "Towards understanding tcp performance on lte/epc mobile networks," in *Proceedings of the 4th Workshop on All Things Cellular: Operations, Applications, & Challenges*, ser. AllThingsCellular '14. New York, NY, USA: ACM, 2014.

[11] I. Johansson, "Congestion control for 4G and 5G access," Internet Engineering Task Force, Internet-Draft draft-johansson-cc-for-4g-5g-02, Jul. 2016, work in Progress.

[12] M. Wang, Z. Zhong, and Q. Liu, "Resource allocation for sc-fdma in lte uplink," in *Proceedings of 2011 IEEE International Conference on Service Operations, Logistics and Informatics*, July 2011.

[13] E. Dahlman, S. Parkvall, and J. Skold, *4G: LTE/LTE-Advanced for Mobile Broadband*. Elsevier Science, 2011. [Online]. Available: https://books.google.fr/books?id=DLbsq9GD0zMC

[14] L. Kleinrock, "Power and deterministic rules of thumb for probabilistic problems in computer communications," in *ICC'79; International Conference on Communications, Volume 3*, 1979.

[15] V. Jacobson, "Congestion avoidance and control," *SIGCOMM Comput. Commun. Rev.*, vol. 18, no. 4, Aug. 1988.

[16] V. P. M. Allman, and and W. Stevens, "Tcp congestion control," NASA Glenn/Sterling Software, Tech. Rep., 1999, rFC2581.

[17] S. Floyd and T. Henderson, "The new reno modification to tcpâĂŹs fast recovery algorithm," ACIRI, Tech. Rep., 1999, rFC 2582.

[18] S. Ha, I. Rhee, and L. Xu, "Cubic: A new tcp-friendly high-speed tcp variant," *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, Jul. 2008.

[19] A. De Vendictis, A. Baiocchi, and M. Bonacci, "Analysis and enhancement of tcp vegas congestion control in a mixed tcp vegas and tcp reno network scenario," *Performance Evaluation*, vol. 53, no. 3-4, 2003.

[20] Y. Zaki, T. Pötsch, J. Chen, L. Subramanian, and C. Görg, "Adaptive congestion control for unpredictable cellular networks," in *ACM SIGCOMM Computer Communication Review*, no. 4. ACM, 2015.

[21] G. Carlucci, L. De Cicco, and S. Mascolo, "Controlling queuing delays for real-time communication: The interplay of e2e and aqm algorithms," *ACM SIGCOMM Computer Commun. Rev*, vol. 46, no. 3, 2016.

[22] X. Xie, X. Zhang, S. Kumar, and L. E. Li, "pistream: Physical layer informed adaptive video streaming over lte," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '15. New York, NY, USA: ACM, 2015.

[23] F. Lu, H. Du, A. Jain, G. M. Voelker, A. C. Snoeren, and A. Terzis, "Cqic: Revisiting cross-layer congestion control for cellular networks," in *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*, ser. HotMobile '15. New York, NY, USA: ACM, 2015.

[24] Z. Zhong, I. Hamchaoui, and R. Khatoun, "Perils of using cqic in lte network and a quick fix with delayed ack," in *15th IEEE Annual Consumer Communications and Networking Conference (CCNC 2018)*, January 2018.

[25] A. Jain, A. Terzis, H. Flinck, N. Sprecher, S. Arunachalam, K. Smith, V. Devarapalli, and R. B. Yanai, "Mobile Throughput Guidance Inband Signaling Protocol," Internet Engineering Task Force, Internet-Draft draft-flinck-mobile-throughput-guidance-04, Mar. 2017, work in Progress.

[26] Z. ZHONG, I. Hamchaoui, R. Khatoun, and A. Serhrouchni, "Performance evaluation of cqic and tcp bbr in mobile network," in *Proceedings 21st Conference on Innovation in Clouds, Internet and Networks (ICIN 2018)*, Paris, France, 2018.

[27] S. S. Chaudhari and R. C. Biradar, "Survey of bandwidth estimation techniques in communication networks," *wireless personal communications*, vol. 83, no. 2, 2015.

[28] B. Veal, K. Li, and D. Lowenthal, "New methods for passive estimation of tcp round-trip times," in *International Workshop on Passive and Active Network Measurement*. Springer, 2005.

[29] S. H. Low, L. L. Peterson, and L. Wang, "Understanding tcp vegas: a duality model," *Journal of the ACM (JACM)*, vol. 49, no. 2, 2002.

[30] M. Hock, R. Bless, and M. Zitterbart, "Experimental evaluation of bbr congestion control," in *2017 IEEE 25th International Conference on Network Protocols (ICNP)*, Oct 2017.