NG-Scope: Fine-Grained Telemetry for NextG Cellular Networks

YAXIONG XIE and KYLE JAMIESON, Princeton University, USA

Accurate and highly-granular channel capacity telemetry of the cellular last hop is crucial for the effective operation of transport layer protocols and cutting edge applications, such as video on demand and video telephony. This paper presents the design, implementation, and experimental performance evaluation of NG-Scope, the first such telemetry tool able to fuse physical-layer channel occupancy readings from the cellular control channel with higher-layer packet arrival statistics and make accurate capacity estimates. NG-Scope handles the latest cellular innovations, such as when multiple base stations aggregate their signals together to serve mobile users. End-to-end experiments in a commercial cellular network demonstrate that wireless capacity varies significantly with channel quality, mobility, competing traffic within each cell, and the number of aggregated cells. Our experiments demonstrate significantly improved cell load estimation accuracy, missing the detection of less than 1% of data capacity overall, a reduction of 82% compared to OWL [9], the state-of-the-art in cellular monitoring. Further experiments show that MobileInsight-based [25] CLAW [39] has a root-mean-squared capacity error of 30.5 Mbit/s, which is 3.3× larger than NG-Scope (9.2 Mbit/s).

CCS Concepts: • **Networks → Transport protocols**; **Cross-layer protocols**; **Physical links**; **Network performance evaluation**; **Mobile networks**.

Additional Key Words and Phrases: Telemetry; Cellular network; NextG system; Capacity estimation; Congestion Control; Video streaming

ACM Reference Format:
Yaxiong Xie and Kyle Jamieson. 2022. NG-Scope: Fine-Grained Telemetry for NextG Cellular Networks. *Proc. ACM Meas. Anal. Comput. Syst.* 6, 1, Article 12 (March 2022), 26 pages. https://doi.org/10.1145/3508032

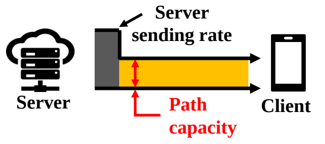
1 INTRODUCTION

Cellular traffic has accounted for 72% of global mobile traffic in the past few years, with this trend expected to continue through 2022 [13]. Motivated by this, network designers are devoting significant efforts to improve the congestion control design of end-to-end transport protocols, so key applications such as video streaming, video telephony [11, 17, 28, 37, 45], and bulk data transfer [6, 10, 15, 16, 20, 46], can work seamlessly with cellular networks, whose channel capacity varies significantly within a very short period of time.

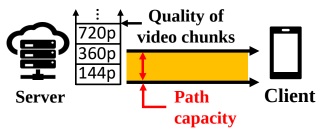
Path capacity, *i.e.,* the capacity of a TCP connection, is a fundamental system parameter that many end-to-end protocols and applications require as an input. For example, to maximize throughput and minimize delay, modern congestion control algorithms, like BBR [10], PCC [15, 16], Copa [6], and PBE [40], try to match senders' rates to path capacity, as shown in Figure 1(a), so high performance requires accurate and fine-grained capacity estimation [40]. Another example is a video streaming application, where the server fragments whole videos into small chunks, encodes each chunk into multiple bit rates, and adaptively selects the bit rate (and thus resolution) of each chunk, according

Authors' address: Yaxiong Xie, yaxiongx@cs.princeton.edu; Kyle Jamieson, kylej@cs.princeton.edu, Princeton University, Princeton, New Jersey, USA.

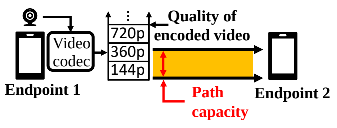
2476-1249/2022/3-ART12
https://doi.org/10.1145/3508032



(a) Congestion control matches the sending rate to path capacity.



(b) The server adapts the streaming video quality to path capacity.



(c) A mobile user adjusts video encoding quality to path capacity.

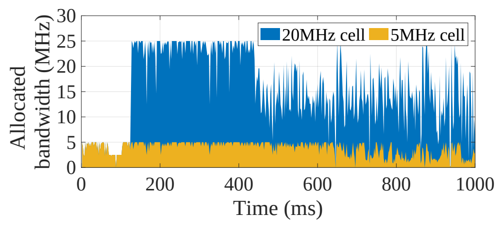
Fig. 1. Congestion control algorithms of the transport layer protocols (**a**); video streaming (**b**); and video telephony (**c**) applications, require accurate and fine-grained channel capacity to achieve a maximized system performance.

to its path capacity estimate and buffer occupancy at the mobile client [5, 21, 28, 34, 44], as shown in Figure 1(b). Unlike video streaming, which pre-encodes the video into all available bit rates, video telephony endpoints encode the video in real time [32, 47]. Consequently, endpoints use available path capacity to adjust their encoded video quality, minimizing video delivery latency and maximizing user quality of experience (QoE) as shown in Figure 1(c).

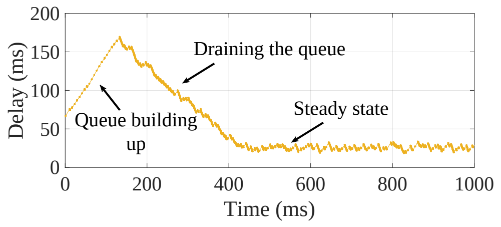
One determinant of the end-to-end path capacity is the capacity of the last-(or first-)hop: wireless cellular Radio Access Network (RAN). Estimating and tracking the capacity of a wireless RAN, is challenging, since its capacity varies over millisecond-level time scales for the following three reasons. First, cellular networks adopt OFDMA to share bandwidth among all mobile users at a subframe (millisecond) granularity. Therefore, the start and termination of any user's data flow can cause abrupt capacity fluctuations. Second, the wireless channel quality between the cell tower and the user varies due to user mobility, multipath propagation, and interference. The highest data rate the cellular wireless channel can support changes accordingly. Third, both 4G LTE [26] and the 5G [4] implement a technique called *carrier aggregation* (CA), via which, the cellular network aggregates two or more cell towers (called *component carriers*) to boost maximum per-user data rates. Because of carrier aggregation, one user is not only affected by the dynamics from one cell but from all of its aggregated cells.

The dynamic nature of a cellular network motivates a telemetry tool that is able to provide accurate and millisecond-granular capacity estimation updates for the cellular network. On the one hand, with the provided accurate capacity estimation, the congestion controller and the upper layer applications can adjust its system parameters, such as send rate or video resolution, with the underlying network condition. On the other hand, with the millisecond-granular capacity updates, the controller and applications can detect network variations with up to millisecond-level delay and thus take quick actions to mitigate the impact of such network dynamics. We note that even though there may exist a delay between the observation of the network variations and the time the actions of the controller or end-to-end applications take effect (the maximum delay is one propagation delay depending on where the observation is taken), (the delay is minimized when the action and the observation are taken at the same place and the delay is maximized to one RTT if the observation has to be delivered back to the sender), our experimental results in Section §5.4 and Section §5.5 demonstrate that such a timely observation still improves the performance of congestion control and video streaming applications significantly.

The opportunity to design such an accurate and fine-grained telemetry tool arises from one observation we make: the cell tower broadcasts all the wireless parameters that affect the wireless cellular channel capacity, including the *modulation and coding rate* (MCS), the number of MIMO spatial streams, the allocated frequency bandwidth, and the aggregated cell towers, to the mobile



(a) Allocated RAN bandwidth.



(b) One way packet delay.

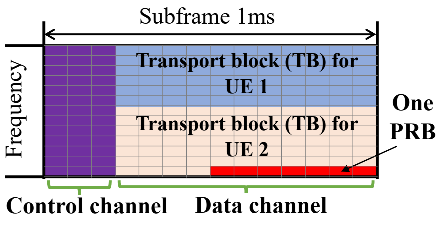
Fig. 2. An example of carrier aggregation. The offered load of the server (25 Mbit/s) exceeds the capacity of the 5 MHz cell, resulting in packet queuing. A cellular network aggregates a 20 MHz cell to boost the maximum capacity available to such a user.

users via a physical control channel, every one millisecond. By carefully decoding this physical control channel, we expose the internal state of the wireless cellular network to transport layer and applications at the endpoints, opening new possibilities for designing agile transport layer protocols and implementing video applications with maximized QoE. Passive cellular sniffers such as LTEye [23] and OWL [9] already decode the control channel to infer some of the above wireless attributes, but cannot work with the most cutting edge cellular networks [4, 26] that uses techniques such as carrier aggregation, and thus are not able to track the abrupt capacity changes the carrier aggregation causes, as the example in Figure 2 shows. MobileInsight [25] can only analyze the radio resource allocation of a single user, *i.e.*, the mobile device the MobileInsight is implemented on, rather than cell-wide information, which is mandatory in order to estimate the capacity available to the mobile user.

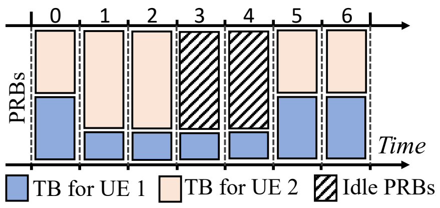
This paper presents the design and implementation of *NG-Scope*, a tool that exposes the fine-grained capacity-related information applications and transport protocols require for superior performance. NG-Scope simultaneously decodes the physical layer control channels of multiple cell towers, extracting highly granular, per-user link and physical layer transmission status information for all users associated with those cells. By combining these data across users within one cell, and fusing data across cells, NG-Scope observes and accounts for the effects of carrier aggregation, estimating wireless cellular channel capacity more accurately than previous cellular monitors. By reconciling control messages from the cellular physical layer with the packet arrival time series from the transport layer of the User Equipment (UE), we accurately monitor the downlink delivery process of every packet the user receives across layers and down to the Physical Layer (PHY), enabling the better functionality of congestion control algorithms and video applications, which heretofore lacked insight into the PHY.

We have implemented a proof of concept prototype of NG-Scope with multiple USRP software-defined-radios, one host PC, and one mobile phone, in a design that synchronizes and fuses information from these sources (§4)[1]. We implement using USRP instead of directly deploying on mobile devices because our design requires customization to the firmware of the cellular module, which is closed-source. However, MobileInsight-based systems like CLAW and BurstTracker that we compare in our evaluation section are directly implementable on mobile devices. Our performance evaluation in a commercial cellular network validates the accuracy and measurement resolution of NG-Scope. NG-Scope's control channel decoder achieves a reduction in missed control messages to 0.8% (a 82% reduction compared to OWL). NG-Scope reduces the rate of false positives (detection of nonexistent control messages) by 92% compared to OWL. Further results show that

[1]The source code of NG-Scope is available at https://github.com/YaxiongXiePrinceton/NG-Scope



(a) LTE subframe structure.



(b) Resource allocation.

Fig. 3. LTE divides time into millisecond-length subframes. The OFDM symbols inside a subframe are allocated to a control channel carrying control information and a data channel carrying transport blocks (TBs) for specific users.

even though wireless capacity varies significantly with mobility, competing traffic within each cell and the number of aggregated cells, NG-Scope is still capable of tracking the resulting capacity variations (§5.3).

We also make new experimental observations on cellular wireless physical- and link-layer operations. Firstly, we find that there exist dedicated cell towers that only function as a secondary cell to deliver carrier-aggregated downlink traffic. Without competition from primary users, such a cell ensures a stable capacity boost for UEs (§5.3), motivating a congestion control that can exploit such an opportunity. Secondly, we observe that with carrier aggregation enabled, the cellular network saturates the primary cell before the potentially higher speed secondary and tertiary cells (§5.7.1), resulting in inefficient bandwidth usage. Lastly, we observe cross-cell traffic correlation (§5.7.2), the effect of competing for traffic propagating to other cells via carrier aggregation.

To further demonstrate the advantage of NG-Scope's complete cell-wide information over the partial information MobileInsight [25] observes, we compare NG-Scope with two systems built atop of MobileInsight: CLAW [39] for capacity tracking (§5.8) and BurstTracker [7] for bottleneck determination (§5.9). Experimental results show that NG-Scope achieves superior performance compared to these two systems.

2 LTE PRIMER

Here we introduce relevant parts of LTE's Layer 1 and 2 data-plane design, focusing on frequency division duplexing (FDD), the mode cellular operators use most widely.

Resource allocation. LTE uses OFDMA in the physical layer [2], dividing the whole channel into smaller time-frequency blocks called *physical resource blocks* (PRB), each spanning 12 OFDM subcarriers in frequency and 500 μs in time, as shown in Figure 3(a). LTE also divides the time into millisecond-length *subframes*, consisting of 14 OFDM symbols in time. Roughly speaking, within the subframe, LTE allocates one to three OFDM symbols to a *control channel* (containing resource allocation information), and the remainder to a *data channel*. Depending on traffic load, LTE allocates PRBs inside the data channel to one or more UEs, with each user's allocation termed a *transport block* (TB). TB size varies across time, depending on the user's traffic load and competition across users, as Figure 3(b) shows.

Retransmission. Wireless transmissions are subject to errors. LTE relies on a Hybrid ARQ (HARQ) mechanism at the link layer to retransmit errored data bits [1]. LTE's HARQ adopts a *stop-and-wait* scheme, sending one transport block in each subframe and then waiting for an acknowledgment from the receiver. If the decoding of a transport block fails, the cell tower resends the erroneous transport block, after eight milliseconds of the original transmission. An explicit *new-data indicator* is included in the control message for every transport block to differentiate the original transmission from the retransmission. One stop-and-wait HARQ process cannot send any data while awaiting an acknowledgment. For continuous data delivery, LTE starts eight HARQ processes that run in parallel, for every associated mobile device.

3 SYSTEM DESIGN

We first introduce NG-Scope's architecture (§3.1), and then the control channel decoder, a crucial component in our system (§3.2). Finally, we detail how we fuse the cross-layer information in our *fusion layer* (§3.3), providing a complete view of wireless cellular communication to higher layers.

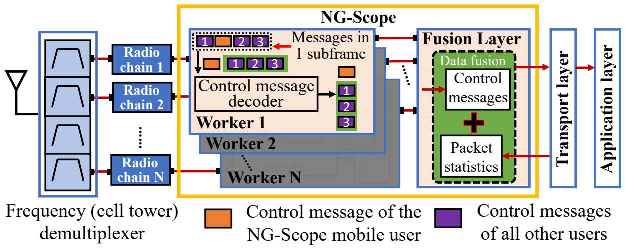


Fig. 4. *NG-Scope architecture.* A customized control message decoder decodes all control messages inside one subframe. A fusion layer fuses control messages from the physical layer and packet statistics from the transport layer.

3.1 System Architecture

A mobile user with carrier aggregation triggered communicates simultaneously with multiple cell towers, as shown in Figure 4. The mobile user starts one worker to decode the control messages from one cell. NG-Scope designs a control message decoder that is capable of decoding all users' control messages in the control channel, as shown in Figure 4. NG-Scope fuses the decoded control messages from the physical layer with packet statistics such as packet arrival time, one-way delay and packet size, from the transport layer, which are reported back to the transport and application layer to facilitate congestion control or video quality selection.

3.2 Control Message Decoder

In this section, we introduce NG-Scope's control message decoder, which is capable of decoding every control message inside one subframe, thus providing a full picture of the bandwidth usage of the cell tower at millisecond time granularity.

3.2.1 Message encoding and transmission inside control channel. In this section, we introduce the background of control message encoding and its transmission via the physical control channel.

Message encoding and decoding. A control message is a bit-string, with every single bit or group of bits inside the message representing various control information, such as the PRB allocation or the MCS index. The length of the bit-string and the exact information each bit conveys depends on the *message format* the base station selects for each message. Base station calculates a 16-bit CRC based on the control message, and then XORs the calculated CRC with another 16-bit physical layer ID of a mobile user (which is the receiver of this message), *i.e.*, the *cell radio network temporary identifier* (C-RNTI), as shown in Figure 5. Base station appends the XOR-ed value at the end of the control message and encodes the message and the appended CRC using convolutional code.

During the decoding process, the appended CRC serves two purposes: message verification and receiver identification. After convolutional decoding, a mobile device separates the 16-bit appended

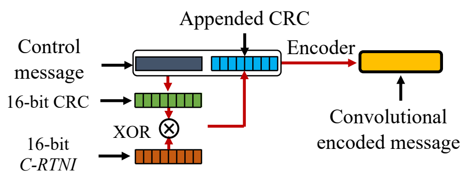


Fig. 5. Control message's encoding process. A 16-bit CRC is calculated from the message, which is XOR-ed with the receiver's ID, *i.e.*, another 16-bit C-RNTI. The XOR-ed value is appended at the end of the packet.

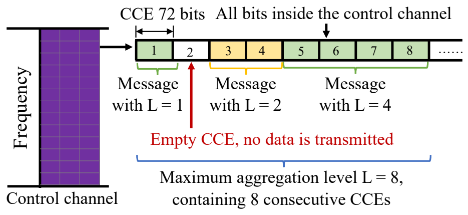
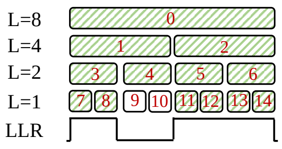

CCE 72 bits
All bits inside the control channel
1
2
3
4
5
6
7
8
Frequency
Message with L = 1
Message with L = 2
Message with L = 4
Empty CCE, no data is transmitted
Maximum aggregation level L = 8, containing 8 consecutive CCEs
Control channel

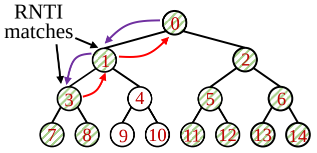
Fig. 6. The distribution of control messages inside the control channel. The bits inside the control channel are aligned in order and then grouped into 72-bit CCEs. Each control message occupies L (aggregation level) consecutive CCEs.

CRC with the control message. To verify the correctness of the decoding process, the mobile device calculates the CRC using the decoded message and then XOR the calculated CRC with its own C-RNTI. When and only when the XOR-ed value matches with the appended CRC, the decoding is successful and and such a mobile device is indeed the intended receiver of the message.

The distribution of control messages inside control channel. The base station transmits the encoded control messages via the physical control channel, as shown in Figure 3(a). Even though the number of bits that each control channel contains varies with the bandwidth of the base station, such a number is much larger than the size of one encoded control message, *e.g.*, the control channel of a 20 MHz base station is capable of transmitting a maximum of 84 control messages. Therefore, to organize the transmission of control messages, the base station aligns the bits of the control channel and groups every 72 consecutive bits into a *control channel element* (CCE), as shown in Figure 6. Each encoded control message occupies L consecutive CCEs, where $L = [1, 2, 4, 8]$ is referred to as the *aggregation level*. We note that the size of the convolutional encoded control message is smaller than one CCE so a message is repeated multiple times if it occupies $L > 1$ CCEs, which provides extra redundancy over the convolutional code and thus extra protection over bit errors. The base station transmits nothing in empty CCEs that contain no control messages.



(a) All the candidate message locations with various aggregation level.



(b) Organizing the search space into the hierarchy of a binary tree.

Fig. 7. **(a)** The candidate message locations within each 8-CCE segment. **(b)** The search space can be organized into the hierarchy of a binary tree.

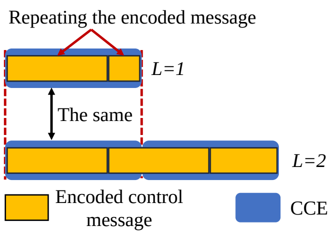


Fig. 8. The first half of a message with L is the same as the whole message with $L - 1$.

3.2.2 *Decoding the whole control channel.*

Typically, one mobile user only decodes its own control message, while NG-Scope has to decode the whole control channel to extract every control message contained inside the control channel. In this subsection, we introduce the challenges we met when trying to achieve that goal and the techniques we propose to address each challenge.

Challenge: the unknowns. The key challenge of designing the decoder is that many important parameters of both the mobile users and the control messages are unknown. Firstly, the physical layer ID of mobile users, *i.e.*, the C-RNTI, are unknown. which is required for both message verification and receiver identification, as we have introduced in Section §3.2.1. Secondly, the total number of control messages contained and their distribution inside the control channel is unknown. At last, for each possible control message, its format and aggregation level are also unknown. Exhaustively searching all possible combinations of location, aggregation level, and format is a possible solution but results in significant computational overhead. To address the above challenges, we propose a tree-based search algorithm and two message validation and user identification algorithms.

Tree-based searching. The locations of the control message are well-organized. Figure 7(a) lists all the candidate locations for eight consecutive CCEs segments, from which we see that a message with aggregation level L has 2^{4-L} possible locations. The whole control channel is divided into multiple such 8-CCE segments. We re-organize the search space of every 8-CCE segment into a binary tree hierarchy, where eight consecutive CCEs represent the eight leaves and the root represents the message that aggregates the eight leaves, as shown in Figure 7(b). The advantage of using a tree-based search is that once any parent node is decoded successfully, we skip searching all of its children nodes since one bit cannot be decoded twice.

We propose to further prune the tree by identifying empty CCEs. The received empty CCE contains no data, only Gaussian noise, which results in significant uncertainty during the demodulation. Therefore, NG-Scope inspects the confidence of demodulating the bits inside a CCE, *i.e.*, the log-likelihood ratios (LLRs), as shown in Figure 7(a), and identifies a CCE as empty if the average LLR of the demodulated bits is below a threshold. NG-Scope identifies empty leaf nodes using LLR and marks a parent node as empty if all of their children are empty. NG-Scope skips searching all identified empty node. We note that the majority of CCEs inside the control channel is actually empty in a commercial LTE network since we have demonstrated in section §5.2 that the base station transmits less than four control messages in more than 99% subframes. Consequently, NG-Scope significantly reduces the computational overhead by identifying and then skipping the empty CCEs.

Deriving the C-RNTI. C-RNTI is necessary for the message validation and receiver identification. As we mentioned in Section §3.2.1 that we are able to separate the message and the appended CRC after decoding, and then calculate the CRC based on the decoded message, as shown in Figure 5. We, therefore, derive the C-RNTI by XOR-ing the calculated CRC and the appended CRC.

Message validation via child-ancestor matching. We introduce our first technique to verify the decoded message and the derived C-RNTI. Our first observation is that the first half of the message with L is exactly the same as the whole message with $L - 1$, just as shown in Figure 8. The fundamental reason behind the observation is that the base station needs to repeat the short encoded message to fill in multiple CCEs, as Figure 8 shows. According to the observation, we know that we will get exactly the same results when decoding the whole message with L and decoding only its first half. We, therefore, traverse the tree shown in Figure 7(b) according to an in-order tree traversal. After a search of one node in the tree, we compare the decoded messages from this node with the messages from its ancestors. If these two messages include the same decoded payload and derived C-RNTI, the decoded message is identified as *validated* and the associated C-RNTI is the ID of a real mobile user. Accordingly, the children of the identified ancestor are removed from the tree.

Message validation via temporal UE tracking. The above children-ancestor matching algorithm cannot handle messages with low aggregation levels. We, therefore, design a validation scheme by

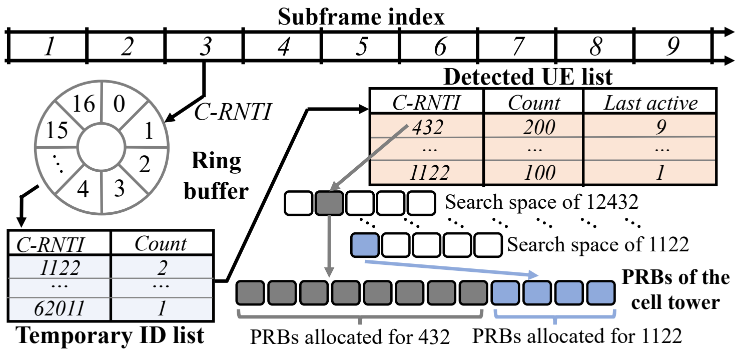


Fig. 9. The NG-Scope *UE Tracker*. NG-Scope buffers recent decoding results, maintaining a list tracking each C-RNTI's recent prevalence.

leveraging the temporal user pattern. Our key observation is that C-RNTIs that reappear within a short period of time (16 *ms* in our implementation) are likely real, as C-RNTIs calculated using incorrect control message parameters (location, aggregation, and format) are random and evenly distributed. The possibility of re-hitting the same C-RNTI generated using wrong parameters can be calculated according to the birthday paradox [8]. As the space of possible C-RNTIs is large in size (2^{16}), the re-hitting possibility is nonetheless extremely small. For example, the possibility that two C-RNTIs reappear within 1,000 messages are 1×10^{-4}.

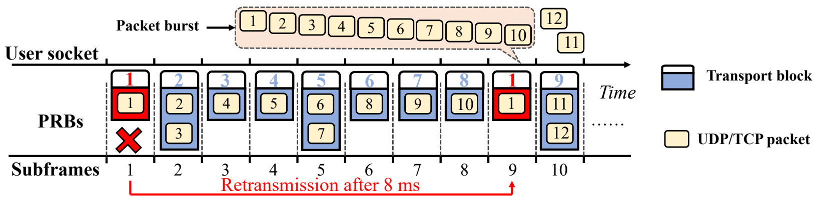
We design a *UE Tracker* to first store the derived C-RNTIs together with the corresponding control messages and track the ID of real UE. The tracker uses a ring buffer to buffer the messages from the 16 most recent subframes, and maintains a *temporary ID list* that stores the count of appearances of each C-RNTI in the ring buffer, as shown in Figure 9. If the C-RNTI count is larger than two, we identify it as the ID of a real UE and move it to the *detected UE* list. The tracker searches the ring buffer for messages associated with C-RNTIs in the UE list, which identified as valid control messages. We record the *last active time* of each C-RNTI (the index of the most recent subframe that this C-RNTI is observed). A C-RNTI that is inactive for 10 seconds is removed from the list.

To reduce the number of buffered messages, we filter out messages with a large number of coded bit errors by re-encoding each decoded control message into coded bits, then comparing the result with the original coded bits inside the received CCEs to calculate the ratio of bits that are erroneous. We drop messages with more than 25% code bits flipped, which is a very high threshold that filters out extremely noisy messages and misses almost no true positive messages, while allowing false positives through, which however are not identified as the ID of real UEs by our UE Tracker.

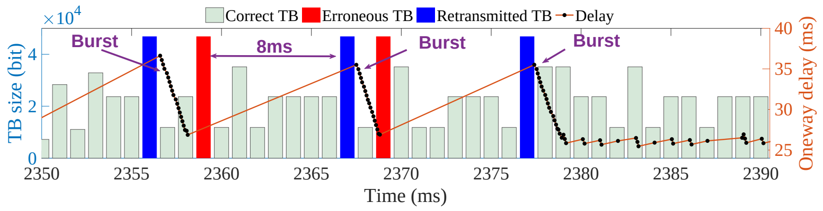
UE tracking with carrier aggregation. When carrier aggregation is active, each aggregated cell tower transmits its own control messages to a UE through its own respective control channel. A UE has only one physical layer ID (C-RNTI), and so the C-RNTI of a UE with carrier aggregation enabled appears in the control messages transmitted by all aggregated cells. NG-Scope extracts the intersected C-RNTIs across the UE lists of all cells, which are identified as having carrier aggregation enabled. NG-Scope identifies the primary and secondary cells of the UE according to the time the ID appears in each aggregated cell, *i.e.*, the earliest cell that the C-RNTI appears in is identified as the primary cell, the second earliest cell is identified as a secondary cell, and so on so forth.

3.3 Data Fusion at Fusion Layer

The transport layer timestamps each received packet. Subframe index naturally provides a one-millisecond resolution timestamp for the control messages contained inside it. The timestamps of



(a) TB loss and reception; packet delivery at the socket.



(b) Received TB size (bits) and one-way delay.

Fig. 10. Packet loss and retransmission interacts with LTE's HARQ design to cause a packet arrival delay of eight milliseconds followed by a burst of packets.

control messages and packet arrivals, however, are unsynchronized related to each other. Therefore, NG-Scope must synchronize and align these two cross-layer information before fusing them.

We leverage the packet arrival pattern of retransmissions to eliminate the shift between timestamps. Figure 10(a) shows an example of TB failure and retransmission, where the UE fails to decode a TB in Subframe 1 that contains one transport layer packet. Eight milliseconds after the original transmission, the cell tower retransmits the lost TB, in Subframe 9. However, as introduced in Section §2, the base station continues transmitting data to the UE between the original transmission and the retransmission. Assuming the TBs sent over subframes two to eight are decoded correctly, the UE receives the second to tenth packets before the reception of the first packet, causing many out-of-order packet receptions. To prevent out-of-order packet reception from happening at a higher layer, the LTE link-layer ensures in-order TB delivery by buffering all out-of-order TBs in a *reorder buffer* until the retransmission recovers the missing TB(s). The link-layer then extracts packets from the TBs and delivers them to the receiving socket, resulting in a burst of packets received at that layer. Furthermore, Figure 10(b) shows that the burst also reshapes the inter-packet-arrival time. The packets inside the burst in fact arrive at the UE uniformly but are instead reported together, creating a large interval between the burst and the packets received before the burst.

From the decoded message, we identify the starting and ending subframe for each retransmission, as shown in Figure 10(a). From the packet log, we detect retransmission based on the eight-millisecond interval between consecutive packets and the packet burst. We, therefore, shift the timestamp of the packet log to match the locations of the retransmissions inside the packet log and the control messages. By doing so, we synchronize the packet log with control messages to millisecond precision, the highest resolution possible since the subframe index is at the same scale.

4 IMPLEMENTATION

Our design is a pure software solution, involving no hardware modifications, so we believe that it can be implemented on a commercial mobile phone by customizing the cellular firmware. The source code of the cellular firmware, however, is proprietary to cellular equipment manufacturers, and is not accessible. We, therefore, implement an open-source proof-of-concept prototype. To emulate multiple radio chains of the commercial phones, we use multiple off-the-shelf software-defined radios, (the USRP [35]) as a front-end to collect cellular signals. The signals the USRP collects are sent to the connected PC for control message decoding. We use a mobile phone that is tethered to the PC for cellular data transmission and reception. Packet statistics and the decoded control messages are fused by the middle layer at the PC.

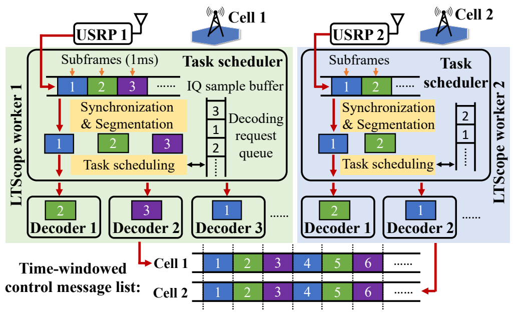
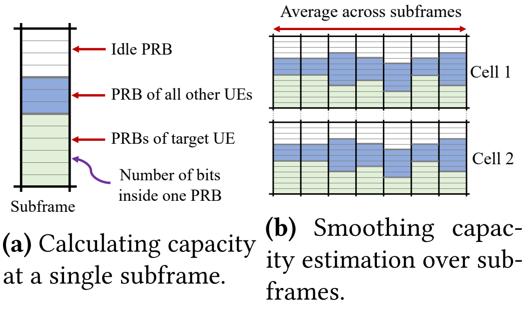


Fig. 11. *NG-Scope's parallel decoding framework.* NG-Scope starts multiple workers in parallel, each of which decodes the control channel of one cell.



(a) Calculating capacity at a single subframe.

(b) Smoothing capacity estimation over subframes.

Fig. 12. Capacity calculation based on information of single subframe **(a)**; Smoothing the capacity estimation over subframes and aggregates the capacity across cells if CA is enabled **(b)**.

We implement a parallel decoding framework as shown in Figure 11. We start one *NG-Scope worker* to handle the signal one USRP streams. Each worker starts one *task scheduler* thread to synchronize with the cell tower and separate the received signal into subframes. The worker also creates multiple *control message decoder* threads, each of which takes the signal of one subframe as input and decodes all control messages inside the subframe. NG-Scope starts more than one decoder threads to guarantee that no matter when a subframe has been separated by the task scheduler, there is a decoder ready to decode it, so the worker buffers a minimum number of subframes, avoiding overflowing the limited socket buffer shared by multiple USRPs. When one decoder thread finishes its decoding task and becomes idle, it sends a decoding request to the task scheduler. The task scheduler allocates decoding tasks according to the order of the decoding requests it receives. The decoded control messages from one worker are ordered according to the index of its subframe and stored in a list. The control message lists of multiple workers are aligned using the subframe index, as shown in Figure 11. We only store the control messages decoded from the most recent 320 subframes.

Capacity estimation. NG-Scope estimates and updates the capacity at the frequency of every one millisecond. After decoding all the control messages of one subframe, NG-Scope knows the PRBs the base station allocations for the target UE and the total PRBs allocated for all other UEs, just as shown in Figure 12(a). NG-Scope calculates the available PRB for target UE as the allocated PRBs for the UE plus the idle PRBs that are not allocated. The base station also tells the UE the number of bits that each PRB can carry, via the control message. Therefore, NG-Scope calculates the capacity of the target UE as the available PRB multiplies the number of bits inside each PRB. To smooth the estimation, NG-Scope averages the available PRB and the number of bits each PRB carries across multiple decoded subframes in the past, as shown in Figure 12(b). If CA is enabled, NG-Scope obtains the overall capacity of the UE by summarizing the capacities of all aggregated cells.

5 EVALUATION

Our performance evaluation quantifies NG-Scope's accuracy and responsiveness in measuring cell load in a head-to-head comparison with OWL (§5.1). We then demonstrate NG-Scope's ability to provide extremely high-granular and accurate cell load estimation in the presence of bit rate adaptation and carrier aggregation, for both stationary and mobile user scenarios (§5.3). We then look at carrier aggregation in-depth, demonstrating unique insights into its operation that NG-Scope enables for the first time (§5.7). At last, we compare with CLAW (§5.8) and BurstTracker (§5.9), two systems built atop of MobileInsight, to demonstrate the advantage of complete cell-wide information NG-Scope decodes from control channel.

#	Configuration	#	Configuration	#	Configuration
1	20 MHz, 1.94 GHz, 2 ant.	2	10 MHz, 739 MHz, 2 ant.	3	10 MHz, 723 MHz, 2 ant.
4	10 MHz, 2.36 GHz, 4 ant.	5	5 MHz, 872 MHz, 4 ant.	6	5 MHz, 1.95 GHz, 4 ant.

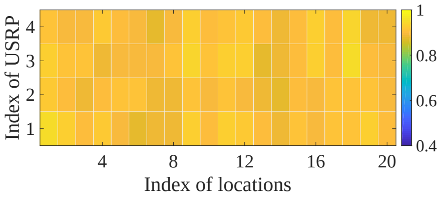
Table 1. Evaluation *cell tower configurations*: frequency bandwidth, center frequency and antenna count.

Experimental configuration. We experiment with the six AT&T cell towers that provide LTE service for the testing area, a university area near a busy street. The detailed configurations of all six cell towers are listed in Table 1. The index of each cell is used to refer to that cell in later sections.

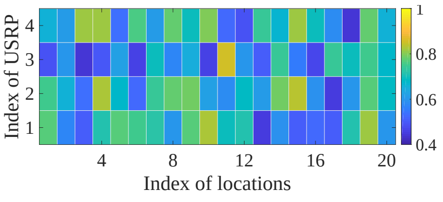
5.1 Decoding Accuracy

In this section, we investigate NG-Scope's decoding accuracy. We compare head-to-head with OWL [9], which has demonstrated superior performance over LTEye [23].

Methodology. Without hacking the cell tower, we cannot get the exact ground truth of control messages the cell tower sends in the control channel of each subframe. To infer the ground truth, we set up four USRPs to listen to the same cell tower at four locations that are one meter apart from each other and apply NG-Scope to decode the received signal. Since the signal received by these four USRPs are uncorrelated, NG-Scope's decoding results using signals from different USRPs are independent of each other. Therefore, messages that appear in the decoding results of all four USRPs are highly likely to be correct and thus are treated as the ground truth of control messages that the cell tower sends. At each location, we repeat the decoding using OWL. We also repeat the entire experiment in 20 combinations of URRP location.



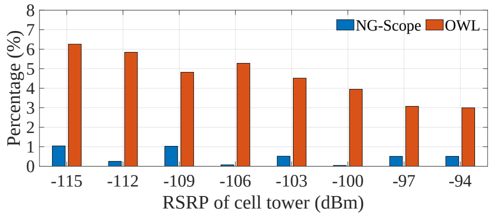
(a) Messages decoded by NG-Scope



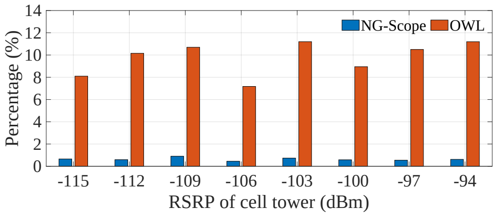
(b) Messages decoded by OWL.

Fig. 13. The percentage of control messages that are correctly decoded by NG-Scope (**a**) and OWL (**b**). We repeat the experiments using four USRPs, across 20 combinations of location.

We calculate the percentage of correctly decoded messages in the decoding results of each USRP and plot the results in Figure 13. NG-Scope's decoding performance is much more stable and accurate than OWL across locations. On average, 90.4% of messages decoded by NG-Scope are correct, while the percentage is 65.3% for OWL. The *true negative*–messages that are missed on any USRP, and *false positive*–messages that are decoded with error and not filtered out, are two main sources of the incorrectly decoded messages. Since we do not have the exact ground truth of every message the base station sends, we cannot get the exact ratio of the true negatives and false positives. We, therefore, conduct the following two experiments to infer these two ratios of NG-Scope and OWL.



(a) Missing data.



(b) False positives.

Fig. 14. Precision and recall of collected packet data **(a)** and generated false positives **(b)**.

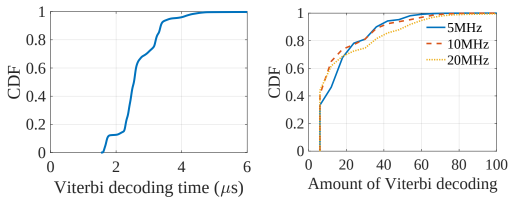
We send 300 Mbit data from the server to a Galaxy S8. We record the amount of data received from the socket of the phone and count the amount of data delivered from cell towers to this phone from the control messages decoded using NG-Scope and OWL. We repeat the experiment at eight locations varying *reference signal received power* (RSRP). We compare the size of data we derive from the control channel with the data received from the socket to get the percentage of data that is missing from the control channel and plot the result in Figure 14(a), We see that NG-Scope captures almost all the UDP traffic sent from the server, with an average missing percentage of 0.83%, a reduction of 83% compare to OWL's 4.5%. Missing control messages results in missing data, so NG-Scope has a smaller percentage of true negative in its decoding results.

We also quantify the generated false positives, a false control message, and thus a nonexistent resource allocation. The total allocated PRBs of one subframe calculated by summing up the allocated PRBs of every control message may exceed the maximum number of PRBs the cell supports, due to false positives. We calculate the percentage of such subframes to infer the generated false positives and plot the results in Figure 14(b). We see that OWL generates a large number of false positives (8%-12%), so that it may overestimate the overall usage of the whole cell. NG-Scope reduces the false positives to the greatest extent possible via its message validation schemes.

5.2 Computational Cost

In this section, we investigate the computational cost of NG-Scope. The basic operation that NG-Scope performs is the convolutional decoding of one control message. Therefore, the computation of each convolutional decoding and the total number of decoding NG-Scope conducts in each subframe determines its computational costs. NG-Scope use the software-implemented Viterbi decoder. Figure 15(a) plots the distribution of the time such a Viterbi decoder takes to decode 3.8 million control messages transmitted by a 20 MHz base station, using an Intel i7-8700 CPU, from which we see that the time is smaller than 4.51 μs for 99% of cases. We note that the hardware Viterbi decoder the mobile phone leverages for convolutional decoding is much faster than our software version.

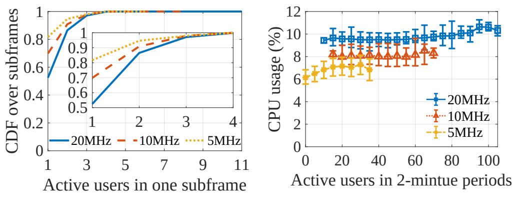
We plot the number of convolutional decoding attempts NG-Scope performs inside each subframe, when decoding the control channel of a 5, 10, and 20 MHz base station, in Figure 15(b). We could observe that NG-Scope performs less than 80 convolutional decoding attempts inside 99% subframes



(a) The CDF of the time each Viterbi decoding consumes.

(b) The CDF of the number of Viterbi decoding performed.

Fig. 15. The time the Viterbi decoder takes to decode one control message **(a).** The number of Viterbi decoding NG-Scope performs to decode all control message inside each subframe **(b).**



(a) The CDF of the number of active users in one subframe.

(b) The number of active users in two-minute time window.

Fig. 16. The number of active users inside the subframes (**a**); the number of user the base station serves within a 2-minute interval and the CPU usage of NG-Scope during each interval (**b**).

of all three base stations. We note that even though a mobile phone only needs to decode its own control message, the phone still needs to blindly perform multiple rounds of convolutional decoding attempts as the mobile phone does not know all the parameters that are required to decode its control message. The mobile phone even does not know whether the base station transmits a control message for it or not, before the blind decoding. According to the standard [3], the maximum decoding attempts each mobile phone needs to perform is 44. Therefore, NG-Scope merely introduces reasonable extra computational cost, *i.e.*, around 1× more convolutional decoding attempts compared with a legacy mobile phone.

The number of active users inside each subframe. The fundamental reason that NG-Scope only introduces limited extra computational overhead is that the number of active users inside each subframe is limited. By active we mean that the base station allocates bandwidth for one user in one subframe and thus transmits one corresponding control message to that user. We derive the number of active users in each subframe by counting the number of decoded control messages and plot the distribution of the number of active users inside every subframe of a base station in 72 hours in Figure 16(a). We clearly see that there are less than four active users and thus less than four control messages in 99.9% of the subframes for all three base stations. Due to the limited amount of control messages to be transmitted, there exist a large number of empty CCEs inside the control channel and thus most of the nodes of the tree in Figure 7(b) are empty, significantly reducing the search space.

The mobile users the base station serves. Even though the base station transmits data to a limited number of mobile devices inside each subframe, the base station serves tens or even hundreds of mobile devices on a longer time scale. To demonstrate that, we divide the 72 hours into 4,320 two-minute periods and plot the number of uses that the base station serves in Figure 16(b), from which we see that a 20 MHz base stations talks to 100 different mobile devices within 2 minutes. We also plot the CPU usage of NG-Scope during each interval. We could see that NG-Scope's computational costs do not increase proportionally with the number of active users the base station serves in a longer period.

5.3 Highly-granular Capacity Tracking using NG-Scope

In this section, we evaluate NG-Scope's highly-granular capacity tracking. We begin with the evaluation of a single cell followed by the aggregated capacity with CA triggered.

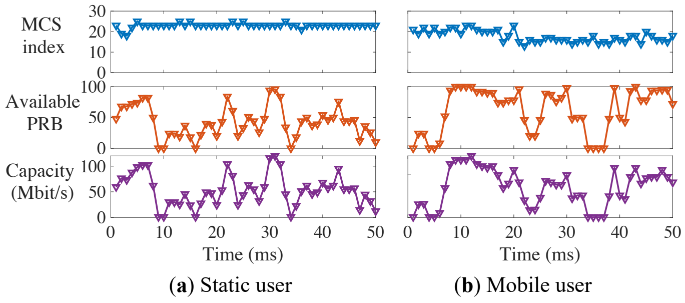


Fig. 17. The estimated wireless link capacity, the decoded modulation and coding rate, and the calculated available PRBs for a static user (**a**) and mobile user (**b**).

Single cell capacity. NG-Scope tracks the LTE wireless link capacity based on the available PRB and the bits each PRB carries for a certain user. The bits each PRB carriers for one use is calculated based on the MCS index and the number spatial stream inside the control message. We plot the MCS index that the one cell selects for the UE (only one spatial stream), the available PRBs, and the calculated LTE link capacity for the static UE at a location with average RSRP −98 dBm, in Figure 17(a). We see that the selected MCS is stable, while the available PRBs of the cell changes dramatically—the capacity varies accordingly. As a comparison, we also plot the same statistics for a mobile UE, in Figure 17(b). We see that, for the mobile user, both the MCS index and available PRBs fluctuate, but at different time scales.

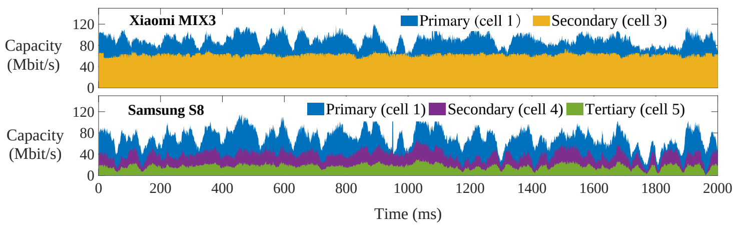


Fig. 18. Aggregated capacities of Xiaomi MIX3 (two cells) and Samsung S8 (three cells). Curves are stacked top to bottom in the order they appear in the legend.

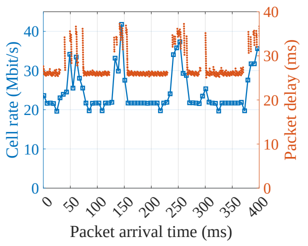


Fig. 19. The LTE physical layer data rate increases with retransmission.

Aggregated capacity. We investigate the aggregated capacity for a certain user with CA triggered. We test with two mobile phones—a Samsung Galaxy S8 and a Xiaomi MIX3. The MIX3 is aggregated with two cells. We plot the aggregated capacity for MIX3 in Figure 18 (*upper*), from which, we see that the capacity provided by the secondary cell is large and stable, while the capacity provided by the primary cell varies significantly. The reason is that the secondary cell operates on LTE band 29, which has only a downlink frequency (generally, one FDD band is divided into uplink and downlink frequency blocks), and so no UE associates with it. This cell is installed purely for carrier aggregation, *i.e.* to work as a secondary cell for downloading data to the UE, so it is idle for most of the time. The S8 is aggregated with three cells. We plot the aggregated capacity for S8 in Figure 18 (*lower*), from which, we see that the capacities provided by three aggregated have large variations. The aggregated capacity varies accordingly.

Impact of TB errors and its retransmissions. We note that not all physical layer capacity is used for transmitting data. When TB error happens, the cell tower needs to allocate bandwidth for both retransmissions of the erroneous TB and ongoing data transmission, resulting in an increase of the instantaneous data rate at the LTE physical layer. To demonstrate this, we plot the LTE physical layer data calculated from the control message and the one-way delay recorded at UE, with a server offered load of 20 Mbit/s. The packet bursts and eight-millisecond interval in one way delay tell us where the retransmission happens, as we have discussed in Section 3.3. We observe that even without retransmissions, the LTE physical layer rate is higher than the offered load (20 Mbit/s) due to protocol overhead (LTE protocol headers). When retransmissions happen, the instantaneous PHY rate increases significantly. The maximum rate can be larger than 40 Mbit/s in this example, which is double the offered load. Since the total PHY capacity of each cell is bounded by the bandwidth, the increasing allocation of capacity for retransmission affects the available capacity for original data transmission, resulting in dynamics of the final available capacity at the transport layer.

5.4 Congestion control using NG-Scope

We implemented a NG-Scope based congestion control algorithm that fully leverages the highly-granular capacity reported by NG-Scope, achieving high throughput and at the same time low latency. By default, our congestion control algorithm sets its sending rate to the capacity reported by NG-Scope. Such a rate causes no congestion when the bottleneck of the TCP connection is at the cellular link. A congestion is detected when the bottleneck shifts from cellular link to the Internet and accordingly our algorithm falls back to a CUBIC algorithm to match its sending rate to the capacity of the Internet link. Our algorithm identifies the bottleneck has shifted back to the cellular link if the rate selected by CUBIC is the same or larger than the capacity reported by NG-Scope.

We measure the performance of our congestion control algorithm in a commercial LTE network (AT&T) and compare its performance with seven other congestion control algorithms, including BBR [10], CUBIC [20], COPA [6], Sprout [37], PCC [15], and PCC-Vivace [16]. We test each algorithm for 30 seconds and repeat the test 20 times for each algorithm. All the performance tests are conducted on workdays when the base station is busy. We measure the performance of the algorithms using Pantheon [43], and plot the averaged achieved throughput and 95^{th} percentile of oneway delay of eight algorithms in Figure 20. We see from the performance results that, with the accurate per-millisecond capacity updates, NG-Scopebased algorithm is able to fully utilize the capacity provided by the cellular network, maximizing its throughput, and at the same time avoid congesting the network, minimizing its latency.

Impact of missing control messages. We investigate the impact of missing control messages on the end-to-end performance of the NG-Scope based congestion control algorithm. In this evaluation, we perform trace-driven emulation using a link emulator: Mahimahi [29]. We decode and record the control messages the base station transmits in a 30 second period, base on which we calculate the available capacity for one mobile user and then feed such a capacity trace to Mahimahi. We build a TCP connection over the link emulated by Mahimahi and feed the recorded control messages to the TCP sender for capacity calculation and congestion control. To emulate the missing messages, we randomly dropped 0% to 50% of the decoded control messages before feeding to the TCP sender.

We measure the achieved throughput and delay of the congestion control algorithm over the emulated link and plot the results in Figure 21. We see that, with the percentage of missing messages increases from 0% to 50%, the achieved throughput increases slightly, *i.e.*, from 39.2 to 39.8 Mbit/s,

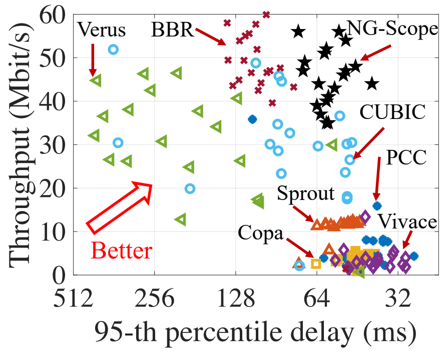


Fig. 20. The average achieved throughput and the 95th percentile oneway delay of eight congestion control algorithms.

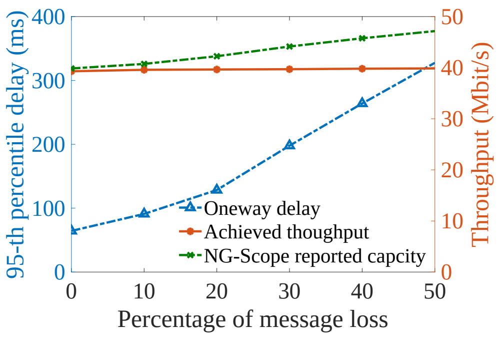


Fig. 21. The average achieved throughput and 95th percentile oneway delay with a varying number of missing control messages.

while the delay increases significantly, *i.e.*, from 64 to 327 *ms*. With more missing control messages, the sender observes more idle bandwidth from the base station and thus over-estimate the available capacity, as shown in Figure 21, causing frequent congestions inside the network.

5.5 Video streaming using NG-Scope

To further demonstrate the value of the telemetry data provided by NG-Scope, we implemented a NG-Scope based video streaming system. MPC [44] is the state-of-the-art video bitrate adaptation system, which estimates the capacity as the harmonic mean of the video downloading speed in the near past. We integrate NG-Scope with MPC [44] by replacing the capacity with the telemetry data measured by NG-Scope, which we refer to as *NG-MPC*. We modify the dash.js to implement our algorithm. We compare our algorithm with MPC [44], Buffer based ABR [21], BOLA [34], and deep learning based ABR – Pensieve [28].

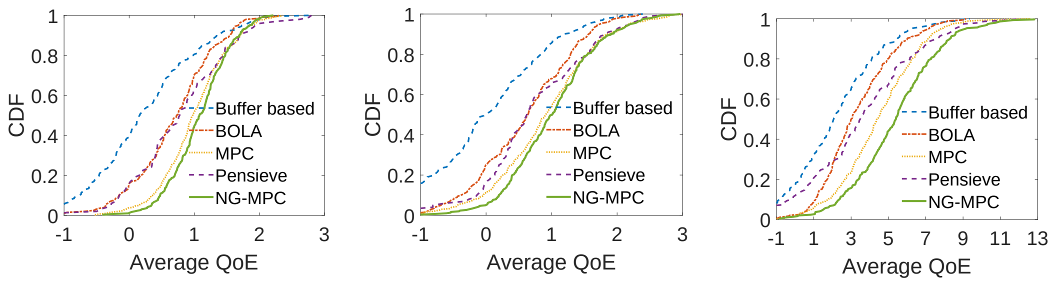
We evaluate the video QoE provided by all five ABR algorithms. There exist a wide range of QoE metrics that characterizes the user-perceived video quality, but three important factors are included in most of the QoE metrics: the average video quality, the quality variations, and the rebuffering. We use the following equation to summarize their impact on the QoE:

$$QoE = \sum_{n=1}^{N} q(R_n) - \mu \sum_{n=1}^{N} T_n - \sum_{n=1}^{N} T_n |q(R_{n+1} - q(R_n)| \tag{1}$$

where N represents the total number of chunks inside a video. The R_n describes the bitrate of $n-th$ chunk and the function $q(R_n)$ translate the bitrate to the user-perceived video quality; the T_n is the rebuffering time when downloading n chunk at bit rate R_n; and the final term characterizes the video bitrate changes which penalizes the overall QoE.

We consider three types of QoE metrics. We start with the first QoE metric QoE_{lin} that has been considered in both MPC and Pensieve, where the quality mapping function is linear $q(R_n) = R_n$. The second metric QoE_{log} is used in BOLA [34] where the quality mapping function $q(R_n) = log(R/R_{min})$, which captures the phenomenon that the improvement of user-perceived quality decreases at higher video bitrates. The third metric QoE_{HD} is used in Pensieve [28] which favors the high definition video by assigning a low score to a low-quality video and a much higher score to a high-quality video. Table 1 in Pensieve [28] gives the detailed score values.

We set up a DASH [14] server and use Google Chrome as the client video player. We stream 10 videos using all aforementioned ABR algorithms. We use Mahimahi [29] to emulate the network



(a) QoE_{lin}. **(b)** QoE_{log}. **(c)** QoE_{HD}.

Fig. 22. Comparing NG-MPC with existing ABR algorithms on the three QoE metrics.

conditions from our cellular traces described in Section §5.4. We measure the final achieve video QoE of different ABR algorithms and plot the results in Figure 22, from which we could observe obvious QoE improvement achieved by NG-MPC over the original MPC. Since we directly use the trained model from Pensieve without retraining it using our cellular traces, we could see that Pensieve cannot provide high quality video streaming as its model does not generalize well, which has also been reported by prior works [5, 42]. With the accurate capacity provided by NG-Scope, the NG-MPC improves the QoE by 7.6%, 10.1% and 11.4% over MPC for QoE_{lin}, QoE_{log}, and QoE_{HD}, over MPC, respectively.

5.6 Tracking Frame Loss and Size using NG-Scope

NG-Scope is able to identify retransmitted transport block using the new-data indicator of the control message. In this section, we investigate how frequently retransmissions happen in commercial cellular networks and their impact on cellular packet transmissions.

Experimental methodology. We use the same setup of the remote server and mobile phone as in §5.1. In our static experiment, we place the mobile phone at one location and let the remote server send UDP packets with a payload length of 1,400 bytes for 10 seconds. We vary the speed of the remote server from 10 Mbit/s to 55 Mbit/s. The phone is connected with the same cell tower (20 MHz bandwidth at 1.94 GHz) during both the static and mobile experiments.

5.6.1 Static user. We move the phone to 10 different locations with varying signal strength in a building. From the decoded control message, we count the number of *original* (not retransmitted) transport blocks that the cell tower sends to the UE. Among all those original transport blocks, we also count the number of transport blocks that have been decoded with bit errors and thus require retransmissions, according to the new data indicator (ndi). We calculate the *TB error rate*, *i.e.*, the ratio of erroneous transport blocks (requiring retransmissions), to the total number of original transport blocks sent. In Figure 23, each row of data comes from a different location and is indexed by the average RSRP at that location. Color represents the value of TB error rate (the lighter colors represent higher TB error rates). We see that the TB error rate varies from 2% to 12%, but there is no obvious pattern of transport block errors across locations. The likelihood of the transport block error at a certain location is determined by the channel at that specific location and how the rate adaptation algorithm works on that channel. We observe however that at each location, the transport block error rate significantly increases with the offered load.

Relationship with TB size. We hypothesize that the increased TB error rate is due to the increased size of the TBs themselves. We plot the distribution of the size of the TBs sent from the cell tower to the UE at the location in Figure 24. We see that with a higher data rate, TB size increases accordingly,

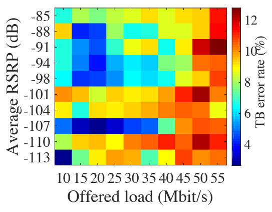


Fig. 23. TB error rate versus offered load and the locations of the user.

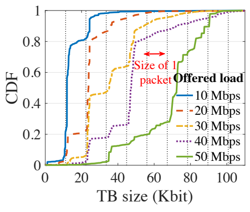


Fig. 24. TB size of TBs sent to the UE with varying offered load.

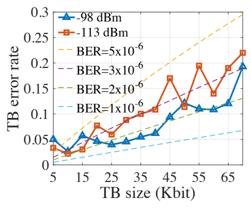


Fig. 25. The TB error rate varies with the size of the TB.

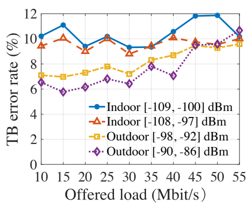


Fig. 26. TB error rate versus offered load for a mobile user.

at a multiple of the packet size, *i.e.*, 1,400 byte. For example, at an offered load of 10 Mbit/s, only one packet arrives at the receiver in each millisecond interval. Accordingly, the cell tower embeds only one packet inside the TB of each subframe, so more than 75% of the TB have a size of only around 1,400 bytes. With increased offered load, more packets are grouped into one TB, increasing its size.

To further validate our hypothesis, we group all received TBs into 14 bins, *i.e.*, zero to 70,000 bits with a step size of 5,000 bits, according to their size. We calculate the TB error rate for blocks in each bin and plot the calculated error rate for locations with RSRP −98 dBm and −113 dBm, in Figure 25. We can see that the TB error rate increases with TB size. We also compare error rates with theory: supposing the error rate of each data bit inside one TB is p and that bit errors are *i.i.d.*, then the TB error rate can be calculated as $1 - (1 - p)^N$, where N is the TB size. We plot the calculated TB error rate for bit error rate p of 5×10^{-6}, 3×10^{-6}, and 1×10^{-6}, in Figure 25. Firstly, we see that the experimental data fit the theoretical predictions, including the *i.i.d.* bit error probability assumption, well. Secondly, we also see that the cellular network maintains the BER for data bits inside the TB at around 10^{-6} and that such a BER is slightly different across locations.

5.6.2 Mobile user. In the mobile experiment, we move the UE along two indoor and two outdoor trajectories at a speed of two m/s. We repeat each trajectory 10 times, with varying offered loads from the server, and record the RSRP range observed when moving along each trajectory. We calculate the TB error rate for each offered load and plot the results in Figure 26. We see that the TB error rate of outdoor trajectory exhibits the same pattern as the static experiments: the higher the offered load, the higher the error rate. But for indoor trajectories, the TB rate is similar for all offered loads. Compared with outdoor trajectories, the channel along the indoor trajectories changes dramatically, due to indoor small-scale fading [33], resulting in more TB errors when LTE rate adaptation algorithms fail to cope with channel variations. We, however, observe that even in the challenging indoor mobile scenario, LTE still manages to keep the error rate below 12%.

5.7 Carrier Aggregation Monitoring using NG-Scope

Accurate determination of congested cell status requires that the UE be aware of the instantaneous carrier aggregation configuration. In this section, we show that NG-Scope can track carrier aggregation in commercial LTE networks.

5.7.1 Carrier Aggregation Load-balancing. Understanding how carrier aggregation affects data delivery provides insights for designing an LTE-compatible congestion control algorithm. Therefore, in this section, we investigate how data are load-balanced over different cells to the UE with carrier aggregation. In this experiment, the UE is static. LTE aggregates three cells (cell one as a primary cell, cell four as secondary, and five as tertiary) for this UE. We let the remote server send UDP traffic at different speeds to the UE for five seconds, then count the number of data bits each cell

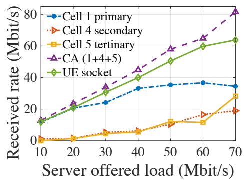


Fig. 27. The received data rate from three aggregated cells and the UE socket.

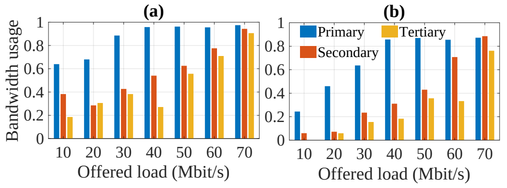


Fig. 28. The ratio of utilized PRB to the total available PRBs (bandwidth usage) of three cells are given in **(a)**; and the bandwidth usages by the UE in three cells are plotted in **(b)**.

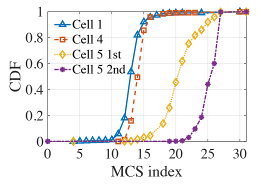


Fig. 29. MCS index CDF for three cells (cell five uses two spatial streams).

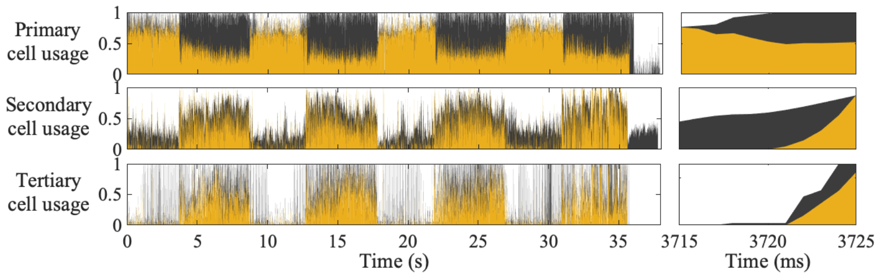
tower delivers to the UE and calculate the average transmitted data rate of each cell. We also record the received data rate from the socket interface of the UE.

We plot the recorded data rates in Figure 27. We have three observations from Figure 27. Firstly, the aggregated rate (summation of three cells) is larger than the received data rate from the UE's socket because of protocol overhead and retransmission of erroneous TBs, which matches with the results in Figure 19. Secondly, the cellular network mainly uses the primary cell for data delivery until it is saturated. To confirm this observation, we also plot the bandwidth usage, *i.e.*, the ratio of allocated PRBs to the total amount of PRBs the cell has, in Figure 28(a) and (b), depicting the bandwidth usage of the cell and the UE, respectively. We see that the primary cell gets saturated (bandwidth usage close to one) earlier (40 Mbit/s) than the other two aggregated cells (70 Mbit/s).

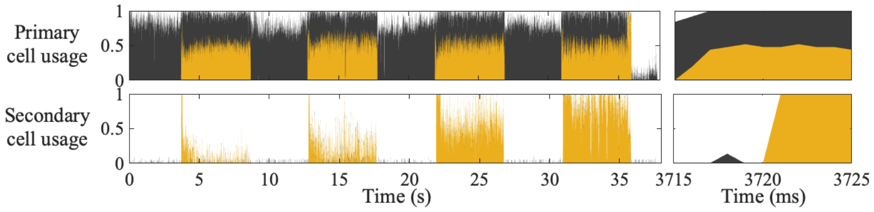
Thirdly, we observe that when all cells are saturated at 70 Mbit/s, cell five with a bandwidth of 5 MHz supports a similar PHY data rate as cell one with a bandwidth of 20 MHz and a much higher data rate than cell four with a bandwidth of 10 MHz. To explain this, we plot the CDF of the MCS index each cell uses to transmit data to the UE in Figure 29. We see that cell five uses two spatial streams and that the MCS used in each spatial stream are much larger than the other two cells, which means that cell five actually has better signal quality and higher spectral efficiency than cell one and five. Such a phenomenon indicates the inefficiency of the LTE load balancing algorithm with carrier aggregation. On the one hand, LTE prefers transmitting data to the UE via the primary cell. On the other hand, the primary cell may not be the one with the highest signal quality within the aggregated cells. Instead of always saturating the primary cell first, a load balancing algorithm that selects the cell for data transmission according to the channel quality of each aggregated cell, would significantly improve overall spectral efficiency.

5.7.2 Competing traffic. In this section, we investigate how competing traffic affects load balancing. In this experiment, we let a server transmit data to Samsung S8 with a constant offered load of 40 Mbit/s. We let another server transmit to a Xiaomi MIX3 with varying offered load from 30 Mbit/s to 60 Mbit/s (each for five seconds) and stop for four seconds between each transmission. The two phones connect to the same primary cell (20 MHz) but different secondary and tertiary cells (if applicable). Specifically, the network aggregates a 10 MHz secondary and a 5 MHz tertiary cell for Samsung S8, and a 10 MHz secondary cell Xiaomi MIX3.

We plot the channel usage of the two phones in these four cells in Figure 30. From Figure 30(a) (*left*), we see that, when there is no competing traffic from the MIX3, most of the data for the S8 are delivered via the primary cell. When the competing traffic starts, the primary cell is saturated. The S8 and MIX3 share that bandwidth, so that a large portion of data for the S8 is offloaded to its secondary and tertiary cells. When the competing traffic is over, all the traffic for S8 is shifted back to the primary cell. We also zoom into the 10 ms period after the starting of competing traffic in Figure 30(a) (*right*). We see that the channel sharing in the primary cell and traffic offloading



(a) Channel usage of primary (cell 1), secondary (cell 4) , and tertiary (cell 5) cells of Samsung S8.



(b) Channel usage of the primary (cell 1) and secondary (cell 3) cells of Xiaomi MIX3.

Fig. 30. The channel usage of the primary, secondary, and tertiary (if applicable) cells of two phones. The black area represents the overall channel usage of the cell. The yellow area represents the usage of mobile phones.

in secondary and tertiary cells are triggered within a few milliseconds, motivating the fine-grain physical layer information that NG-Scope provides. On the other hand, we see from Figure 30(b) that since the primary cell is shared with the S8, increasing the offered load for MIX3 only increases the channel usage in the secondary cell.

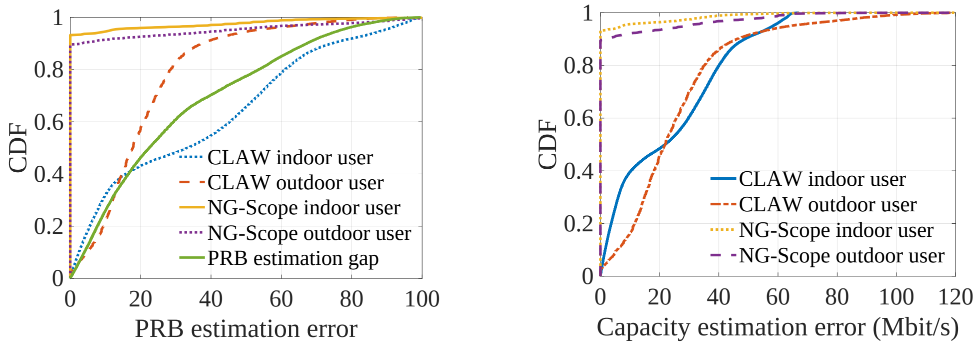
From this experiment, we see that the impact of competing traffic in one cell could propagate to other cells via carrier aggregation. Such a cross-cell traffic correlation makes the resource allocation of each cell highly dynamic, resulting in significant link capacity variations. With the fine-grain information provided by NG-Scope, the UE can track these variations and make responsive actions.

5.8 Capacity Tracking with CLAW

In this section, we compare the accuracy of NG-Scope's capacity tracking with CLAW [39]. CLAW implements a capacity tracker based on MobileInsight [25]. Different from NG-Scope, MobileInsight reports the PRB allocation for a single user, not the full PRB usage of the entire base station. CLAW, therefore, proposes to estimate the PRB usage based on the power measurements reported by MobileInsight, with the intuition that higher utilization results in higher received power. We evaluate the performance of CLAW's PRB estimation and capacity estimation.

Methodology. We put two mobile users associated with the same base station (20 MHz and 100 PRBs) at one indoor and one outdoor location. We implement CLAW on both mobile devices and log the estimation of the PRB utilization of every subframe. We put two NG-Scopes co-located with CLAW users for comparison. We also deploy another three NG-Scopes at three different locations to estimate the PRB usage of the same base station and extract the PRB usage of subframes where four NG-Scopes have exactly the same decoding results, as our ground truth.

We plot the distribution of CLAW's PRB estimation error in Figure 31(a). We see that CLAW has a much larger error in estimating utilized PRBs than NG-Scope. NG-Scope is accurate in estimation



(a) PRB estimation error. **(b)** Capacity estimation error.

Fig. 31. The PRB estimation error of a pair of indoor/outdoor CLAW users is plotted in (**a**); the error of the capacity calculated using the inaccurate estimated PRB is given in (**b**).

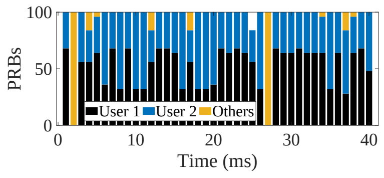
for 92% and 88% subframes for outdoor and indoor users respectively. CLAW, however, has a median and 95-percentile error of 17 PRBs and 38 PRBs for the outdoor user, and 33 PRBs and 74 PRBs for the indoor user, respectively. The underlying reason for such a large estimation error is that the power measurements are vulnerable to interference of neighboring base stations that operate at the same frequency band and thus become quite noisy. Consequently, the PRB estimation based on such a noisy power report is unreliable. Furthermore, CLAW works better if all the PRBs have similar received power at the mobile user. Such an assumption is invalid in an indoor scenario due to the frequency selective fading, so we observe a much higher estimation error for the indoor user. At last, we calculate the gap between two CLAW users' PRB estimations for the same subframe and plot the distribution of the gap across subframes in Figure 31(a). We see that, due to the noise in the power measurements, two CLAW users almost never get the same utilized PRB estimation for the same subframe of the same base station.

We also translate the error of PRB estimation into the error of capacity estimation and plot the results in Figure 31(b). We see that, on average, CLAW has an RMS capacity error of 31 Mbit/s, which is 3.3× of NG-Scope (9.2 Mbit/s). The indoor CLAW user has a larger PRB error but a similar capacity error due to its lower signal strength.

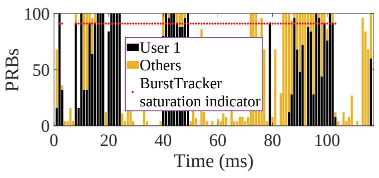
5.9 Bottleneck Detection with BurstTracker

In this section, we compare NG-Scope's performance with BurstTracker on one task: detecting whether the cellular link is the bottleneck of a connection. The cellular link becomes the bottleneck when all the bandwidth of the base station is fully utilized. BurstTracker is a dedicated tool built atop of MobileInsight for such a task. MobileInsight reports only the PRB allocation for the mobile device it is implemented on, so it cannot be directly applied to determine the bottleneck. To bypass such a constrain, BurstTracker makes a hypothesis about the base station's resource allocation algorithm–cell tower allocates all the bandwidth to one user in one subframe, evoking the TDMA. We, however, observe different phenomena in our experimental result. Specifically, from Figure 16(a), we see that a 20 MHz base station serves and thus allocates PRBs to more than one user in 47.5% of the active subframes.

Based on such a hypothesis, BurstTracker identifies the start of cellular wireless link becoming a bottleneck, when MobileInsight reports more than 90% of PRB occupation by this user. To investigate how this heuristic works in practice, we let two users download bulky data simultaneously and plot the PRB allocation results of every subframe, measured by NG-Scope, in Figure 32(a). We clearly see that these two users almost equally share the 100 PRBs of the base station, and there is no subframe in which one user occupies more than 90% of the PRBs. On the other hand, almost all the PRBs of this cell are fully utilized, so the cellular wireless link is indeed the bottleneck for both users. We feed such a trace to BurstTracker (we use the author's implementation) and find that BurstTracker detects no saturation because the utilization of each user stays around 50%.



(a) Two competing users.



(b) One user web-browsing.

Fig. 32. Resource allocation results measured using NG-Scope. **(a)** The cell shares the PRBs inside each subframe (represented by a single bar) among multiple backlogged users, so BurstTracker [7] detects no saturation when it in fact exists; **(b)** when the user is web browsing, BurstTracker indicates long periods of saturation even when the link is not fully saturated.

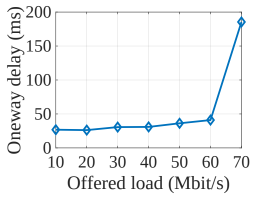


Fig. 33. The achieved average oneway packet delay, with offered load varying from 10 Mbit/s to 70 Mbit/s.

BurstTracker's method to identify the end of wireless saturation relies on a relatively slow decreasing traffic load, *i.e.*, a subframe with less than 40% PRBs allocated for the user, immediately followed by an empty subframe for that user. We, however, observe that the end of wireless saturation in significant amount of traces do not exhibit such a traffic pattern. For example, Figure 32(b) depicts the PRB allocation allocated for one web browsing user along time. We see from this figure that BurstTracker tells us that there exists a long saturation period that lasts for around 100 ms. But if we check the PRB allocation results, we see that the cell tower has multiple idle intervals during the saturation period indicated by BurstTracker. BurstTracker erroneously identifies those idle intervals as saturated because the PRB allocation for the user does not end as required by BurstTracker. Specifically, the PRBs allocated for the user abruptly become zero after a subframe with more than 90% of PRBs allocated.

Detecting the wireless link saturation is an easy task for NG-Scope with the overall cell bandwidth usage decoded from the control channel. For example, we see from Figure 28(a) that three aggregated cells are saturated with offered load 70 Mbit/s so that the LTE wireless link becomes the bottleneck. We confirm this by checking the average oneway delay, which is plotted in Figure 33. We see that the average delay is constantly small with offered load from 10 Mbit/s to 60 Mbit/s. The average delay suddenly increases to 185 *ms*, which is caused by the packet buffering at the base station.

We run BurstTracker with 200 traces when the user is conducting different activities, including file downloading, video streaming, and web browsing, and record the length of the saturation period in each trace. We use the channel usage measured from NG-Scope as ground truth. According to our results, 42% of the time BurstTracker identifies as saturated are truly positive and the rest 58% are false positive. BurstTracker cannot identify the start and end of LTE link saturation correctly and robustly, due to its invalid hypothesis of all-or-nothing resource allocation, and its requirement of a slowly decreasing traffic load. While these heuristics sometimes do function as intended, the fundamental reason that forces BurstTracker to rely on such heuristics is that they can only extract the PRB allocation results for the current user, and not the resource usage of the whole cell.

6 RELATED WORK

Congestion control for cellular network. The key challenge of designing congestion control for cellular network is to estimate and track the rapidly varying capacity. A large body of prior work rely on end-to-end packet statistics to infer link capacity [6, 15, 16, 30, 36, 41]. End-to-end measurements cannot track the fast-varying cellular link capacity, so that these algorithms either under-utilize network bandwidth and/or introduce excess delays. On the other hand, CLAW [39] and piStream [38] adjust the transmission speed according to the base station's bandwidth usage, which is inferred from the received power measurements. Power measurements, however, are vulnerable to interference and thus result in significant estimation error (§5.8). CQIC [27] adjusts its congestion window purely based on the wireless channel quality derived from the *channel quality indicator* (CQI) reported by UE, and thus can only track the capacity variations caused by fluctuations in channel quality. ABC [18, 19] and throughput guidance [22] proposes to redesign the cellular cell tower such that the cell tower can estimate the link capacity of each user and feed this information back to the sender, which requires modifications to the infrastructure and is not compatible with the deployed commercial LTE network. PBE-CC [40] uses a similar method of capacity estimation as NG-Scope, but provides no evaluation on the computational cost and estimation accuracy. NG-Scope, however, fully describes the design and implementation details, conducts extensive evaluation, and performs head-to-head comparison with the state-of-the-art, *i.e.*, OWL [9] and MobileInsight-based CLAW [39].

Video streaming and video telephony. The capacity of the end-to-end connection is required by the adaptive bitrate algorithms (ABR) [5, 21, 34, 44] of video streaming applications and video codec [12, 32, 47] of videotelephony applications, to perform real-time video resolution selection. These systems either rely on the capacity reported by the transport layer protocols [15, 16, 41] or directly derive the capacity based on coarse-grained historical statistics about video delivery at the application layer, and thus cannot track the fast varying capacity in cellular networks.

LTE monitoring tools. LTEye [23] and OWL [9] are two passive LTE sniffers. LTEye does not decode control information for MIMO transmission, while OWL cannot work with a cellular network that implements carrier aggregation. Furthermore, both LTEye and OWL use bit errors inside a control message as an indicator to conduct message validation, which is vulnerable to channel fading and interference, and thus introduces a huge amount of false positives (§5.1). QXDM [31] and MobileInsight [25] are tools that measure only the PRB allocation of a single user, but not the bandwidth usage of the whole cell, resulting in significant errors in capacity estimation. Systems built atop of MobileInsight, including CLAW [39], PERCEIVE [24] and BurstTracker [7] suffers from the same limitation.

7 DISCUSSION AND FUTURE WORK

Telemetry at base station. NG-Scope performs network telemetry at the mobile devices. An alternative solution could be collecting the millisecond-granular capacity information at the base station. We note that, for any end-to-end applications, the base station is a third-party, so the telemetry data collected by the base station cannot be trusted without proper authentication. Building an authentication system between the base station and every end-to-end application that runs atop of it involves significant overhead. Transmitting the telemetry data from the base station back to the mobile devices, however, incurs tremendous communication overhead. For example, if we would like to update the capacity at the same frequency as NG-Scope provides, then the base station has to send 1000 messages to each mobile user every second. Even if we lower the frequency to every 20ms (which results in delayed reactions), the base station still needs to transmit 50 messages per second per user. Considering the number of UE the base station serves, the total number of messages the base station sends will be quite large. Base on the above analysis, we argue in this paper that performing telemetry at the mobile devices is a better choice, at the cost of slightly increased computation overhead for each mobile user, as we have demonstrated in Section §5.2.

Time division duplexing (TDD). The current version of NG-Scope focuses on decoding the control channel of the cellular network that adopts FDD in its physical layer. The main difference between TDD and FDD is the frame architecture. Specifically, FDD uses separate frequencies for uplink and downlink channels, while TDD leverages the same frequency band for both uplink and downlink. We note that the structure of downlink subframes where the physical control channel resides is identical in TDD and FDD. Therefore, NG-Scope is directly applicable to TDD for decoding control messages and then monitoring the capacity. We leave the extension of NG-Scope to TDD as our near-term future work.

Evolving cellular architecture. The design of the physical control channel changes with the evolving of the cellular network architecture, from 4G LTE to 5G and 5G beyond. For example, the physical control channel of 5G NR and 4G LTE differs from the following three aspects: firstly, the 5G NR encodes the physical control messages using polar codes instead of convolutional code; secondly, the location of the control channel inside each subframe does not follow the configuration of 4G LTE as shown in Figure 3(a); thirdly, the formats of the control message, *i.e.,* the meaning of each bit inside a message, are different for 4G LTE and 5G.

We, however, note that two important design choices of the physical control channel still remains in the 5G NR and will keep staying in the 3GPP standards for the foreseeable future. First of all, the 5G NR does not encrypt the control messages they transmit so that each mobile user is capable of decoding all the control messages if they choose to. Secondly, the 5G NR carries the ID of the mobile devices, *i.e.*, the C-RNTI, inside the control message via XOR-ing it with the CRC, as we have introduced in Section §3.2.1. According to the above analysis, with necessary customization to cope with the changes in the 5G control channel, NG-Scope is able to decode all the control messages inside the control channel of the 5G network and associate each decoded message with its ID. Based on the decoded 5G control messages, NG-Scope is, therefore, capable of estimating the 5G network capacity, just as we have demonstrated for 4G LTE.

8 CONCLUSION

NG-Scope sets new benchmarks for accuracy in mobile cellular network monitoring hence enables the development of dramatically improved congestion control algorithms and applications that respond more effectively to fluctuations in the wireless channel. In the course of its design and implementation, we have additionally documented interesting and important experimental phenomena surrounding carrier aggregation that are significant to any network designer considering the traffic that flows through a mobile cellular network.

9 ACKNOWLEDGMENTS

We thank the anonymous reviewers and our shepherd for their valuable feedback that has improved the quality of this paper significantly. This material is based upon work supported by the National Science Foundation under Grant No. CNS-2027647. We gratefully acknowledge a gift from the Amateur Radio Digital Communications Foundation.

REFERENCES

[1] 3GPP. TS 36.321: Evolved Universal Terrestrial Radio Access (E-UTRA); Medium Access Control (MAC) protocol specification.

[2] 3GPP. TS36.211: Evolved Universal Terrestrial Radio Access (E-UTRA); Physical channels and modulation.

[3] 3GPP. TS36.213: Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures.

[4] 5G specifications. Available here.

[5] Z. Akhtar, Y. S. Nam, R. Govindan, S. Rao, J. Chen, E. Katz-Bassett, B. Ribeiro, J. Zhan, and H. Zhang. Oboe: Auto-tuning video abr algorithms to network conditions. In *ACM SIGCOMM*, 2018.

[6] V. Arun and H. Balakrishnan. Copa: Practical delay-based congestion control for the internet. In *NSDI*, 2018.

[7] A. Balasingam, M. Bansal, R. Misra, K. Nagaraj, R. Tandra, S. Katti, and A. Schulman. Detecting if lte is the bottleneck with bursttracker. In *ACM MobiCom*, 2019.

[8] Birthday problem. Wiki.

[9] N. Bui and J. Widmer. OWL: A reliable online watcher for lte control channel measurements. In *ACM AllThingsCellular*, 2016.

[10] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson. Bbr: Congestion-based congestion control. *Queue*, 14(5), Oct. 2016.

[11] J. Chen, R. Mahindra, M. A. Khojastepour, S. Rangarajan, and M. Chiang. A scheduling framework for adaptive video delivery over cellular networks. In *ACM MobiCom*, 2013.

[12] Webrtc. Official website.

[13] Cisco: Global mobile data traffic forecast update. Available here, 2019.

[14] Dynamic adaptive streaming over http. Website.

[15] M. Dong, Q. Li, D. Zarchy, P. B. Godfrey, and M. Schapira. PCC: Re-architecting congestion control for consistent high performance. In *NSDI*, 2015.

[16] M. Dong, T. Meng, D. Zarchy, E. Arslan, Y. Gilad, B. Godfrey, and M. Schapira. PCC Vivace: Online-learning congestion control. In *NSDI*, 2018.

[17] S. Fouladi, J. Emmons, E. Orbay, C. Wu, R. S. Wahby, and K. Winstein. Salsify: Low-latency network video through tighter integration between a video codec and a transport protocol. In *NSDI*, 2018.

[18] P. Goyal, A. Agarwal, R. Netravali, M. Alizadeh, and H. Balakrishnan. ABC: A simple explicit congestion controller for wireless networks. In *USENIX NSDI*, 2020.

[19] P. Goyal, M. Alizadeh, and H. Balakrishnan. Rethinking congestion control for cellular networks. In *ACM HotNets*, 2017.

[20] S. Ha, I. Rhee, and L. Xu. CUBIC: A new TCP-friendly high-speed TCP variant. *SIGOPS Oper. Syst. Rev.*, 42(5), July 2008.

[21] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In *ACM SIGCOMM*, 2014.

[22] A. Jain, A. Terzis, H. Flinck, N. Sprecher, S. Arunachalam, K. Smith, V. Devarapalli, and R. Yanai. Mobile throughput guidance inband signaling protocol. *IETF, work in progress*, 2015.

[23] S. Kumar, E. Hamed, D. Katabi, and L. Erran Li. LTE radio analytics made easy and accessible. In *ACM SIGCOMM*, 2014.

[24] J. Lee, S. Lee, J. Lee, S. D. Sathyanarayana, H. Lim, J. Lee, X. Zhu, S. Ramakrishnan, D. Grunwald, K. Lee, and S. Ha. Perceive: Deep learning-based cellular uplink prediction using real-time scheduling patterns. In *ACM MobiSys*, 2020.

[25] Y. Li, C. Peng, Z. Yuan, J. Li, H. Deng, and T. Wang. Mobileinsight: Extracting and analyzing cellular network information on smartphones. In *ACM MobiCom*, 2016.

[26] LTE Release 10. Available here.

[27] 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 *ACM HotMobile*, 2015.

[28] H. Mao, R. Netravali, and M. Alizadeh. Neural adaptive video streaming with pensieve. In *ACM SIGCOMM*, 2017.

[29] R. Netravali, A. Sivaraman, S. Das, A. Goyal, K. Winstein, J. Mickens, and H. Balakrishnan. Mahimahi: Accurate record-and-replay for http. In *USENIX ATC*, 2015.

[30] S. Park, J. Lee, J. Kim, J. Lee, S. Ha, and K. Lee. Exll: An extremely low-latency congestion control for mobile cellular networks. In *ACM CoNEXT*, 2018.

[31] Qualcomm qxdm tool. Website.

[32] D. Ray, J. Kosaian, K. V. Rashmi, and S. Seshan. Vantage: Optimizing video upload for time-shifted viewing of social live streams. In *ACM SIGCOMM*, 2019.

[33] S. Sesia, I. Toufik, and M. Baker. *LTE, The UMTS Long Term Evolution: From Theory to Practice.* Wiley Publishing, 2009.

[34] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman. BOLA: Near-optimal bitrate adaptation for online videos. In *IEEE INFOCOM*, 2016.

[35] USRP. Website.

[36] K. Winstein and H. Balakrishnan. TCP Ex Machina: Computer-generated congestion control. In *ACM SIGCOMM*, 2013.

[37] K. Winstein, A. Sivaraman, and H. Balakrishnan. Stochastic forecasts achieve high throughput and low delay over cellular networks. In *USENIX NSDI*, 2013.

[38] X. Xie, X. Zhang, S. Kumar, and L. E. Li. piStream: Physical layer informed adaptive video streaming over lte. In *ACM MobiCom*, 2015.

[39] X. Xie, X. Zhang, and S. Zhu. Accelerating mobile web loading using cellular link information. In *ACM MobiSys*, 2017.

[40] Y. Xie, F. Yi, and K. Jamieson. PBE-CC: Congestion control via endpoint-centric, physical-layer bandwidth measurements. In *ACM SIGCOMM*, 2020.

[41] Q. Xu, S. Mehrotra, Z. Mao, and J. Li. PROTEUS: Network performance forecast for real-time, interactive mobile applications. In *ACM MobiSys*, 2013.

[42] F. Y. Yan, H. Ayers, C. Zhu, S. Fouladi, J. Hong, K. Zhang, P. Levis, and K. Winstein. Learning in situ: a randomized experiment in video streaming. In *USENIX NSDI*, 2020.

[43] F. Y. Yan, J. Ma, G. D. Hill, D. Raghavan, R. S. Wahby, P. Levis, and K. Winstein. Pantheon: The training ground for internet congestion-control research. In *USENIX ATC*, 2018.

[44] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli. A control-theoretic approach for dynamic adaptive video streaming over HTTP. In *ACM SIGCOMM*, 2015.

[45] J. Yoon, H. Zhang, S. Banerjee, and S. Rangarajan. MuVi: A multicast video delivery scheme for 4G cellular networks. In *ACM MobiCom*, 2012.

[46] Y. Zaki, T. Pötsch, J. Chen, L. Subramanian, and C. Görg. Adaptive congestion control for unpredictable cellular networks. In *ACM SIGCOMM*, 2015.

[47] A. Zhou, H. Zhang, G. Su, L. Wu, R. Ma, Z. Meng, X. Zhang, X. Xie, H. Ma, and X. Chen. Learning to coordinate video codec with transport protocol for mobile video telephony. In *ACM MobiCom*, 2019.

Received October 2021; revised December 2021; accepted January 2022