Real-time Bandwidth Prediction and Rate Adaptation for Video Calls over Cellular Networks

***Abstract*—We study interactive video calls between two users, where at least one of the users is connected over a cellular network. It is known that cellular links present highly varying network bandwidth and packet delays. If the sending rate of the video call exceeds the available bandwidth, the video frames may be excessively delayed, destroying the interactivity of video call. In this paper, we present Rebera, a joint cross-layer design of proactive congestion control, video encoding and rate adaptation to maximize the video transmission rate while keeping the one-way frame delays sufficiently low. Rebera actively measures the available bandwidth in real-time by employing the video frames as packet trains. Using an online linear adaptive filter, Rebera makes a history-based prediction of the future capacity, and determines a rate budget for the video rate adaptation. Rebera uses the hierarchical-P encoding to provide error resilience and easy rate adaptation, while maintaining low encoding complexity and delay. Furthermore, Rebera decides in real-time whether to send or discard an encoded frame, according to the budget, thereby preventing self-congestion and minimizing the packet delays. Our experiments with real cellular link traces demonstrate Rebera can deliver higher bandwidth utilization and shorter packet delays than Apple's FaceTime.**

I. Introduction

Advances in networking and video encoding technologies in the last decade have made the real-time video delivery applications, including video calls and conferencing [1][2][3], an integral part of our lives. Despite their popularity in wired and Wi-Fi networks, real-time video delivery applications have not found much use over cellular networks. The fundamental challenge for delivering real-time video in cellular networks is to simultaneously achieve high-rate and low-delay video transmission over the highly volatile cellular links with fast-changing available bandwidth, packet delay and loss. On a cellular link, increasing the video sending rate beyond the available bandwidth leads to self-congestion, and intolerable packet delays, and consequently frame delays. Overly delayed frames will have to be treated as lost. On the other hand, a conservative sending rate clearly leads to the under-utilization of the cellular channel and consequently a lower quality than what is possible.

The extremely tight design space calls for a joint cross-layer design of realtime video encoding, and bitrate control at the application layer, and sending rate adjustment, and error control at the transport layer. Ideally, one would like the transmitted video rate closely keep track of the capacities on cellular links. However, the traditional *reactive congestion control algorithms* [4], [5] that adjust sending rate based on congestion feedbacks, in forms of packet loss and/or packet delay, are too slow to adapt to the changes in capacity, leading to either bandwidth under-utilization or long packet delays [6]. It is more preferable to design *proactive congestion control algorithms* that calculate sending rate based on forecasting cellular link capacities. Meanwhile, for video adaptation, video encoder can adjust various video encoding parameters, so that the resulting video bitrate matches the target sending rate determined by the congestion control algorithm. However, accurate rate control is very challenging for low-delay encoding, and significant rate mismatch is often still present with the state-of-the-art video encoders. In addition, what makes the problem even more challenging is that lost and late packets can render not only their corresponding frames but also other frames non-decodable at the receiver. The encoder and the transport layer should be designed to be error resilient so that lost and late packets have minimal impact on the decoded video.

In this study, we propose a new real-time video delivery system, Rebera, designed for cellular networks, where we aim to maximize the sending rate of the video source and error resilience, while keeping the one-way frame delays sufficiently small. Our system consists of a proactive congestion control module, a temporal layered encoder, and a dynamic frame selection module. Our proactive congestion control module uses the video frames themselves to actively measure the current available bandwidth in real-time, and then use the well-known linear adaptive filtering methods [7] to predict the future capacity, based on the past and present capacity measurements. For error resilience, we resort to layered encoding, which enables unequal error protection (UEP). However spatial and quality layering incurs significant

encoding complexity, making them unattractive for practical employments. Thus we consider only temporal layering, which provides a certain level of error resilience even without using explicit UEP. To minimize the delays for real-time delivery, we use hierarchical-P (hP) coding structure for temporal layering. To address the rate control inaccuracy of the encoder, we propose a dynamic frame selection algorithm for hP, where the goal is to select in real-time which encoded frames to send, subject to a bitrate budget determined by the predicted capacity value. Our frame selection algorithm takes into account quality implications and decoding dependency between frames, and the smoothness of frame inter-arrivals to maximize the delivered video quality under the bitrate budget. We implement the complete system, which we call Rebera for real-time bandwidth estimation and rate adaptation, on a testbed and we evaluate the performance of the system and compare it with Apple's FaceTime video call application. Our current implementation relies on a off-line encoder that produces a fixed average rate video. Our experiments show that, even in the absence of a real-time encoder that can directly control the encoded video rate according to the measured capacity, Rebera is able to achieve higher bandwidth utilization and lower frame delays than FaceTime. The current study has not considered UEP among the temporal layers and the error resilience aspect of the system. These will be investigated in future studies.

A. Related Work

Authors of [6] proposed a proactive congestion control scheme for realtime video delivery in cellular networks. They model cellular links as single-server queues emptied out by a doubly-stochastic service process. For the available bandwidth estimation, we, unlike [6], we assume no particular time-evolution model for the link capacity. Furthermore, [6] only focused on congestion control without considering video adaptation. As will be shown in our work, joint design of video adaptation and congestion control is crucial to achieve high quality in video delivery in cellualr networks.

II. Proposed System Overview

We examine a real-time video delivery scenario between a sender and a receiver, where at least one user is connected to a cellular network (Fig. 1). We denote the source device by S, the destination device by D, and the corresponding base stations by B_S and B_D, respectively. We assume that the in-network path (B_S, B_D) that connects the base stations have higher available bandwidth, and constant queuing and propagation delay.

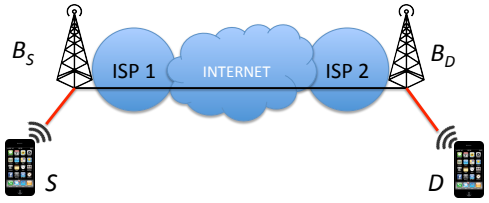

Fig. 1. Cellular links between the mobile devices and their respective base stations are in red. ISPs are mobile operators.

Therefore, the overall available bandwidth along the path (S, B_S, B_D, D) is equal to the minimum of the bandwidths along the cellular links (S, B_S) and $B_D, D)$.

According to the queuing model of [6], all packets destined to or sent from a given mobile device that is connected to a base station are queued up in isolated buffers, which are located on the mobile device for the uplink and in the base station for the downlink. These buffers are not shared by any other flow to or from other users; that is, there is no cross-traffic in these queues. The backlogged packets leave their respective buffers once they are successfully transmitted over the link. Thus, how fast these buffers are emptied out directly reflects the capacity of the cellular links, and consequently the end-to-end (e2e) available bandwidth.

As for the video stream, we assume that the sender uses a layered encoder so that it can easily adjust the sending rate by adjusting the number of video layers transmitted. Layered coding also enables unequal error protection; i.e., a basic level of quality can be guaranteed with high likelihood by providing more protection to the base layer. We consider only temporal scalability to keep the encoding complexity and overhead at a minimum. In order to minimize the encoding delay, we further assume that the hP structure (Fig. 3) is used to achieve temporal scalability. Starting with the highest temporal layer, the frames can be discarded to reduce the video rate. In the example shown in Fig. 3, each Group of Picture (GoP) consists of 4 frames, which leads to three temporal layers (TLs). We assume that the encoder inserts an I-frame every N frames, and we denote the time duration covering all N frames from an I-frame up to but excluding the next one by an intra-period. Then, the time duration T for an intra-period is equal to N/f, where f is the frame rate of the captured video.

We can now summarize the operation of the proposed system. Since rate control is usually performed once per intra-period in conventional video encoders, we predict the average cellular capacity for each new intra-period. The prediction is based on the measured average capacities for the past intra-periods, which are measured and fed back to the sender by the receiver. In order to circumvent the uncertainty about the possible

feedback delay, the receiver periodically measures (every Δ seconds, $\Delta \ll T$) the available bandwidth within the last T seconds using the arriving video frames and feeds these measurements back to the sender. The sender, in turn, keeps the most recent capacity measurement, and updates its value with the arrival of each new measurement. Then, at the beginning of the next intra-period k, the value of the most recent capacity measurement is taken as the available bandwidth $\tilde{c}_{k-1}$ measured during the last intra-period $k-1$. This value is input to an adaptive linear prediction filter, which then updates its prediction $\hat{c}_k$ regarding the available bandwidth during the new intra-period k using the past bandwidth values $\tilde{c}_{k-1}, \ldots, \tilde{c}_{k-M+1}$. Using this prediction, the sender calculates the rate budget b_k, which is the maximum number of bytes that the sender is allowed to send during this intra-period so that all the data sent make it to the receiver with a high probability at the end of the intra-period. The components of our design can be seen in Fig. 2.

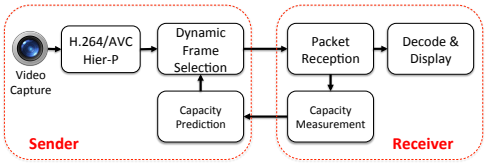

Fig. 2. Proposed real-time video delivery system over cellular networks

III. Capacity Measurement & Prediction and Determination of the Sending Rate

A. Measuring the End-to-End Available Bandwidth

Packet pair/train methods [8] are well-known active capacity measurement schemes for finding the minimum capacity along a network path. The performance of these methods improve if there is no cross-traffic on the links, making them suitable to measure the cellular link capacity according to our model. In our system, we propose measuring the average available bandwidth $c(t_1, t_2)$ actively at the destination, using the video frames received in $(t_1, t_2]$ as packet trains. Using the video frames as packet trains enables us to directly exploit the video data flow for capacity measurements and to avoid sending additional measurement traffic. Specifically, at the sender side, we first packetize each frame regardless of its size into $p \geq 2$ packets , and then send them together in a burst. The resulting instantaneous sending rate is likely to be higher than the instantaneous capacity of the link. As a result, packets queue up in the bottleneck; i.e. the base station buffer for the downlink or the cellular device buffer for the uplink, where they are transmitted one by one. Then, at the receiver side, we take capacity measurements $\{m_n\}$, where the sample m_n is obtained by using the arriving video frame n as a packet train. Let us denote the inter-arrival time between packet $i-1$ and i by a_i, and the size of the packet i by z_i. Then, we can calculate the capacity sampled by frame n as:

$$m_n \triangleq \frac{z_2 + \cdots + z_p}{a_2 + \cdots + a_p} \triangleq \frac{Z_n}{A_n}. \tag{1}$$

For any time period $(t_1, t_2]$, we can estimate the average capacity $c(t_1, t_2)$ over this time simply by

$$\tilde{c}(t_1, t_2) = \frac{\sum_{n \in \mathcal{N}} Z_n}{\sum_{n \in \mathcal{N}} A_n}, \tag{2}$$

where $\mathcal{N}$ is the set of all frames that arrived in $(t_1, t_2]$. Note that Eq. (2) is equivalent to taking a weighted average of all the capacity samples in $\{m_n\}$, where the sample m_n is weighted in proportion to its measurement duration A_n with weight $w_n = A_n / \sum_{n \in \mathcal{N}} A_n$. Having completed the average capacity measurement regarding $(t_1, t_2]$, the receiver prepares a small feedback packet and sends it back to the source. Note that we are ultimately interested in measuring the available bandwidth c_k during $(T_k, T_{k+1}]$, where T_k denotes the start of the intra-period k. However, since the sender and receiver have different clock times in general, the receiver cannot know when exactly an intra-period starts. Furthermore, the feedback packets are subject to time-varying delays in the network. In short, we cannot guarantee that the feedback packets will arrive at the sender on time for predicting the capacity of the next intra period. To address this issue, the receiver measures the average capacity within the last T seconds every Δ seconds, where $\Delta \ll T$. Each of these measurements are immediately sent back to the sender. Specifically, a measurement generated at time t is the average capacity in $(t-T, t]$, while the next measurement that is generated at $t+\Delta$ is the average bandwidth in $(t-T+\Delta, t+\Delta]$. The sender then uses the latest feedback received before T_k to predict the available bandwidth in the next intra-period $(T_k, T_{k+1}]$. Lastly, assuming we keep the sending rate below the capacity, our measurement accuracy depends on the difference between the sending rate and the capacity of the link. If the sending rate equals, or by any chance, exceeds the capacity, we would have very high measurement accuracy, but this may lead to saturated links and long queueing delays, which are detrimental to video call quality.

B. Predicting the End-to-End Available Bandwidth

History-based forecast is a popular method for prediction [9], where the past measurement values are used

to determine an estimate of the future. In this study, we perform linear prediction for history-based forecast. In particular, we chose a well-known online linear adaptive filter called the Recursive Least Squares (RLS) [7]. With each new capacity measurement regarding the last intra-period, RLS recursively updates its filter taps of length M, and makes a prediction for the capacity during the next intra-period. One of the advantages of the RLS algorithm is that it does not require a model for its input signal, and performs the minimum least-squares regression [10]. Furthermore, it can adapt to the time-varying signal statistics through its forgetting factor λ, which serves to exponentially discount the weight of the past observations, without any need for a time-evolution model.

Robustness against bursts: It is known that the cellular links occasionally experience channel outages that may last for up to several seconds, during which the capacity essentially drops to zero, and the packets in transit are backlogged in the respective buffers. As a result, the sender should stop sending any more packets as soon as an outage is detected. When the outage ends, all the packets queued up in the buffer are usually transmitted and arrive at the receiver as a burst. If the receiver uses these packets for capacity measurement, the burst rate, which is on the order of several Mbps, can severely disrupt the learning process of the predictor. In order to protect our system against these bursty measurements, we simply detect them through the sample measurement duration A_n. In our system, we consider a measurement bursty if $A_n < 10$ ms. Bursty measurements are simply discarded.

C. Determining the Sending Rate

Our ultimate goal is to ensure that all the frames sent during an intra-period finish their transmission before the start of the next one. In other words, we aim to have each I-frame encounter empty buffers with high probability. Let us denote our sending rate in the intra-period $k+1$ by r_{k+1}. We determine r_{k+1} such that the probability to exceed the capacity c_{k+1} is low; that is,

$$\Pr(c_{k+1} < r_{k+1}) = \delta, \tag{3}$$

where δ is a small confidence parameter. Let ϵ_{k+1} denote the capacity prediction error obtained from the RLS algorithm, i.e, $c_{k+1} = \hat{c}_{k+1} + \epsilon_{k+1}$, we can rewrite Eq. (3) as

$$\Pr\left(\epsilon_{k+1} < r_{k+1} - \hat{c}_{k+1}\right) = \Pr\left(\epsilon_{k+1} < u_{k+1}\right) = \delta, \tag{4}$$

where $u_{k+1} \triangleq r_{k+1} - \hat{c}_{k+1}$ is referred to as safety margin. This means that, for a given δ value, r_{k+1} should be set as the sum of $\hat{c}_{k+1}$ and u_{k+1}, the δ-quantile of ϵ_{k+1}. In Rebera, we set $\delta = 0.05$, and calculate the running 5-percentile of ϵ_{k+1} with a moving window [11].

Handling backlogged and lost packets: Note that sending at a rate always less than the available bandwidth cannot be guaranteed, even with the safety margin u_k. Left unaddressed, the number of bytes backlogged in the buffers may grow indefinitely. In order to address this issue, we could measure the number q_k of bytes backlogged in the buffers at the end of intra-period k by subtracting the total number of bytes received at the receiver from the total number of bytes sent at the sender. However, in case of packet losses, since lost packets will never be seen by the receiver, q_k would keep growing in time. In order to account for the losses, we assume that the packets arrive at the destination in the order of their sequence numbers. To detect the number of lost bytes, we insert in each packet header the total number of bytes sent so far. Then, upon receiving a new packet, the receiver simply subtracts the number of bytes it has received so far from this number. The result is the number of bytes lost, which is fed back to the sender, along with the number of bytes received. The sender then determines q_k by taking the difference between the total number of bytes sent and the total number of bytes received and lost.[1]

Combining all, we set the bandwidth budget b_{k+1} for the intra-period $k+1$ as

$$b_{k+1} = (\hat{c}_{k+1} + u_{k+1})T - q_k, \tag{5}$$

where T is the intra-period duration. *This way, we expect the network not only can finish the transmission of all video frames in intra-period $k+1$, but also can clean up the currently backlogged packets q_k by the end of intra-period $k+1$.*

IV. Frame Selection for Hierarchical-P Video

Video rate control is crucial for real-time applications over networks with time-varying bandwidth. However, accurate rate control is very challenging, especially in the very low-delay scenarios, where look-ahead and multi-pass encoding are not suitable. In spite of the extensive research in this area [12], significant mismatch between the target and actual bitrate over an intra-period can still occur [12]. In case of rate mismatch, if the video is coded with the IPPP structure, all remaining frames will be

[1] Out-of-order packet deliveries will introduce only temporary errors to our estimates: after the delayed packets arrive at the receiver, our algorithm will automatically correct these errors in the next estimate.

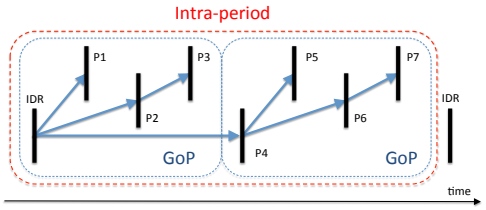

Fig. 3. hP prediction. Blue arrows indicate the reference frames used to predict the frames being coded. In this example, $N = 8$, $G = 4$, TL0: (I0,P4); TL1: (P2,P6); TL2: (P1, P3, P5, P7).

discarded once the target budget for an intra-period is used up. When this happens early in the intra-period, the receiver will experience a relatively long freeze.

To remedy this problem, we propose to use a temporal layered encoder with the hP coding structure, so that the sending rate can be adjusted by skipping the higher layer frames, without incurring the additional encoding delay or complexity. Fig. 3 shows an example prediction structure for the hP encoding, which yields three temporal layers: I-frame and P4 form the base layer, P2 and P6 form the second layer, and P1, P3, P5, P7 form the third layer. We propose a frame selection scheme that either discards or sends each encoded frame, subject to the given rate budget b_k and frame dependencies. Note that here we assume that video encoder runs its own rate control algorithm, but may not meet the rate budget per intra-period accurately. When the encoded bitrate exceeds the budget, an encoded frame may be dropped by the frame selection scheme so that the actual sending rate never exceeds the predicted bandwidth for an intra-period. The benefit of using the hP structure is that the delivered video has more evenly-spread frames, whereas the IPPP structure can lead to a very jittery video when some frames are dropped. With the frame selection module outside the encoder, the encoder rate control can be less conservative. This, in turn, can lead to higher bandwidth utilization.

A. Dynamic Frame Selection

Frame selection is ultimately about allocating the budget for more important, i.e., lower layer frames. Higher layer frames can be sent only if there is available rate budget after sending the lower layer frames, depending on the budget. However, to minimize the delays, the decision to either send or discard a given frame must be made right after it is encoded, without knowing future frame sizes. For example, in Fig. 3, we cannot wait to see if we can send P4 first, followed by P2 and then P1. Rather, we have to determine whether we send P1 as soon as P1 arrives. If the future frames from lower layers are large, sending frames from a current higher layer may preclude the sending of upcoming lower layer frames. On the other hand, dropping frames from higher temporal layers when the future lower layer frames are small would clearly underutilize the available bandwidth.

Given an intra-period, let us label each frame with its appearance order, and denote the size and the temporal layer of the frame n by s_n and ℓ_n, respectively. Our goal is to decide, for each encoded frame n, to either send or discard it, such that the total number of frames sent at the end of the intra-period is maximized, while the mean and the variance of the time gap between the selected frames are kept small. We start our frame selection procedure by estimating frame size for each temporal layer, in order to make decisions considering the future frames. We then continue by ordering the frames in this intra-period based on their layer numbers, starting with the lowest layer, since the higher layer frames cannot be decoded with the lower layers. We denote this priority order by an ordered list π. For each newly arriving frame n, we trim π into π_n by excluding the past frames for which a decision has already been made, and the future frames that cannot be decoded at the receiver due to the previously discarded frames. π_n is basically the priority order among the eligible frames left. Next, we update the frame size estimations, as well as our estimation for the remaining rate budget. Then, we create a set E_n of frames that we expect to send according to our frame size and the remaining rate budget estimations, by greedily picking frames starting from the first frame in π_n. We stop picking the frames when the total estimated size of the frames picked reaches the estimated rate budget. Finally, if frame n is in the set E_n, we send it; otherwise it is discarded.

For frame size estimation, we assume that the frame sizes within the same temporal layer will be similar. Therefore, we keep a frame size estimate $\hat{s}_\ell$ for each temporal layer ℓ. In this study, we simply use the exponentially weighted moving average (EWMA) filter with parameter $0 < \gamma \leq 1$ for estimating the size of future frames in layer l using actual sizes of the past coded frames in this layer. Note that for the base layer, we apply the above method only to the successive P-frames as the I-frame size is much larger than P-frames. We do not need to estimate the I-frame size, as we always send the I-frames. The overall algorithm is summarized in Algorithm 1.

B. Rate Budget Update

The rate budget b_k is the estimation of the total number of bytes that the sender can transmit during the intra-period k without causing buffer build-up. Here, we assume that, at any time t since the start of the intra-

period, $\frac{t}{T}b_k$ bytes can be transmitted on average, with a mean rate of b_k/T. Thus, if the sender sends less than this amount, the unused bandwidth is wasted. In order to account for these missed transmission opportunities, we update the remaining rate budget at each step n by

$$\hat{b}_k(n) = b_k - \max\left(S_n, \frac{n}{N}b_k\right), \tag{6}$$

where S_n is the total number of bytes sent before selecting frame n. Without updating the budget, the sender may end up sending large frames close to the end of the intra-period, which would then backlog in the buffer, and potentially delay all the packets in the next intra-period.

Algorithm 1 Dynamic Frame Selection

```
S_0 = 0, π_0 = π, intra-period k, rate budget b_k
for all frames n = {0, ..., N − 1} do
    ŝ_j ← γ s_n + (1 − γ) ŝ_j, for each frame j ∈ ℓ_n
    b̂_k(n) = b_k − max(S_n, (n/N) b_k)
    Create E_n from π_n, based on ŝ and b̂_k(n)
    if n ∈ E_n then
        S_{n+1} = S_n + s_n and send frame n
    else
        π_{n+1} = π_n − {frames depending on n}
    end if
    π_{n+1} = π_n − n
end for
```

V. Simulations and Experiments

A. Adaptive Filtering vs. Exponential Smoothing for Bandwidth Prediction

We start our evaluations by motivating the use of the RLS linear adaptive filter for capacity prediction. We compare the prediction performance of the RLS with the simple and popular EWMA predictor [9]. In our experience, the filter length and the forgetting factor parameters do not significantly affect the prediction errors provided that we choose $M < 10$ and $\lambda > 0.99$. Therefore, we have selected $M = 5$, $\lambda = 0.999$, $\theta = 0.001$ and fixed this configuration for the rest of the evaluations. We collected eight real cellular link capacity traces (Fig. 7) following the methodology in [6], using 3G and HSPA access technologies, during different times of the day and in different campus locations. Each of these is $1,066$ seconds long and their statistics can be found in Table I. As expected, the capacity traces are very dynamic, posing significant challenge to capacity estimation. Over these traces, in Matlab, we perform time-series forecasting using RLS with parameters mentioned above, and the EWMA filter, where the smoothing parameter α is varied from 0 to 1. We assume that we know the past capacity values exactly. The results can be seen in Table II, where "Best" and "Worst" represent the minimum and maximum prediction error root-mean square (RMS) values obtained with EWMA with different smoothing parameters, respectively. We see that for all traces, prediction performance of RLS is very close to that of the best EWMA predictor, if not better, as it adapts to the statistics of the capacity time series.

TABLE I
Statistics of cellular traces used in the experiments.

	Mean (kbps)	Std (kbps)	Coeff. of Var.	Outage %
Tr1	176	115	0.654	2.0
Tr2	388	165	0.425	0.5
Tr3	392	202	0.514	5.2
Tr4	634	262	0.413	0.0
Tr5	735	264	0.359	0.2
Tr6	937	356	0.379	1.2
Tr7	1055	501	0.475	0.1
Tr8	1564	893	0.571	5.1

TABLE II
Comparing prediction error RMS of RLS with EWMA

	RLS (kbps)	α_B	Best (kbps)	α_W	Worst (kbps)
Tr1	53	0.55	55	0.05	87
Tr2	88	0.7	90	0.05	120
Tr3	87	0.65	86	0.05	132
Tr4	158	0.55	157	0.05	209
Tr5	186	0.4	178	0.05	211
Tr6	250	0.2	235	1	293
Tr7	244	0.4	242	0.05	291
Tr8	894	0.1	858	1	1212

B. Dynamic Frame Selection Simulations

We now compare the performance of our dynamic frame selection (DFS) algorithm against the layer-push (LP) and frame-push (FP) algorithms. LP also estimates the frame size in each temporal layer using the same approach as in DFS, but then decides on the highest layer $l_{\max}$ that may be sent. In other words, only the frames in layers up to $l_{\max}$ are eligible for sending. Among these frames, following the encoding order, the algorithm sends as many frames as possible until the rate budget is exhausted. FP, on the other hand, does not consider layer information and sends as many frames as possible following their encoding order, until the rate budget is exhausted.

For each algorithm, we consider the total number of frames sent, the mean and standard deviation values of the resulting frame intervals, and finally the fraction of the unused rate budget. To calculate the statistics of the frame intervals, we use the fractions of time that the

receiver will observe the frame intervals. We use the JM encoder [13] to encode the video sequence "Crew" [14] with a hP structure having three temporal layers (GoP length=4) and intra-period of 32 frames. We used a fixed quantization parameter (QP) of 36, yielding the average bitrate of 415 kbps when all frames are included. The resulting video sequence has a frame rate of 30 fps and comprises 9 intra-periods, with an intra-period of $T = 32/30$ seconds. For the proposed algorithm, we used $\gamma = 0.75$, which was found to perform the best, and the frame priority order is $\pi = (0, 4, 8, 12, 16, 20, 24, 28, 30, 14, 6, 22, 26, 18, 10, 2, 31, 15, 7, 23, 27, 19, 11, 3, 1, 5, 9, 13, 17, 21, 25, 29)$. In these simulations, we assume that the rate budget b_k is constant for each intra-period k of the video and we want to compare the performance of algorithms described above under different b_k values, from 10 kB to 80 kB. In Figure 4, we see that FP sends the most frames by sending as many frames as possible. However, it also has the highest mean frame interval and frame interval variation, making the displayed video jittery. On the other hand, the LP algorithm sends the lowest number of frames but also with lower mean frame interval and frame interval variance. The proposed DFS algorithm achieves a good compromise between sending more frames, consequently utilizing available bandwidth more closely, and reducing the frame distance variation. In fact, DFS outperforms both methods in terms of the mean and standard deviation of the frame intervals, while sending almost as many frames as the FP. Finally, the plot in the upper right shows the fraction of the unused bandwidth for each method, where we see that the performance of DFS is very similar to FP, whereas LP is not as efficient.

C. Evaluation of Rebera and FaceTime on Testbed

For system evaluation, we developed a testbed to compare Rebera with popular video call applications. On this testbed (Fig. 5), S and D are the source and destination end-points running the video call application under test, while the nodes C_S and C_D are cellular link emulators running the CellSim software [6], respectively. The emulators are connected to each other through the campus network, and to their respective end-points via Ethernet. For cellular link emulation, we use the uplink and downlink capacity traces collected as described in Table I. For evaluation, we use the available bandwidth utilization and the 95-percentile one-way packet queuing delays as the performance metrics. In order to calculate the bandwidth utilization, we count how many bytes were sent out by the video call application under test and and compare it with the minimum of the capacities of the

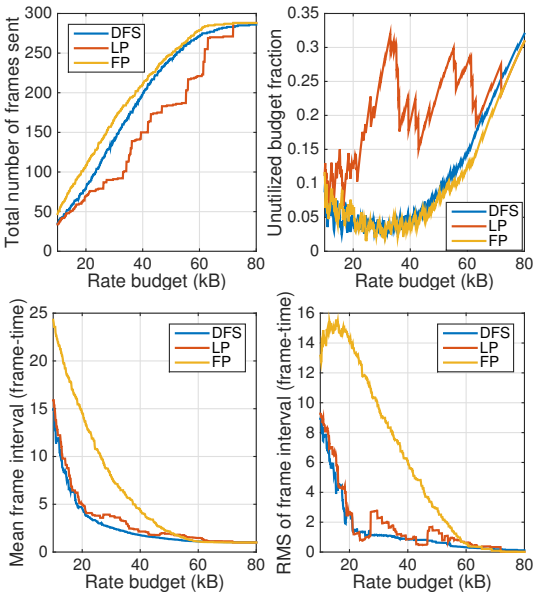

Fig. 4. Comparison of DFS with FP and LP with respect to number of frames sent (upper left), unused budget (upper right), mean and standard deviation of the frame distance (lower left and lower right)

sender link and the receiver link. The queuing delays are collected through CellSim for Rebera experiments, and by sniffing the video packets on the emulator machines for FaceTime. This is because FaceTime sends voice packets even after the voice is muted, and we would like to measure the delays regarding the video packets, only. In each test, we use the same video sequence "Crew" by looping it periodically. While testing Rebera, we transmit a pre-encoded video stream using the JM encoder [13] as described earlier, since we lack a video encoder capable of producing a hP stream in real-time. The result is that the frame sizes are fixed for Rebera, whereas the commercial products can further adapt the encoding parameters to tune the frame sizes according to the capacity. The video and RLS parameters used are the same as in Sections V-B and V-A. The initial sending rate is set to 120 kbps. In each experiment, we evaluate the sending rate over consecutive periods of T seconds. Please note that FaceTime may not be using a constant intra-period, let alone the same intra-period T as Rebera. Furthermore, FaceTime's sending rate is, in general, the sum of FEC and the video data rates. In order to feed the same looped test video into FaceTime, we used the ManyCam [15] virtual webcam on Mac OS 10.10.4.

1) Evaluation under Piecewise Constant Bandwidth: In this experiment, we use a piecewise constant bandwidth trace, with steps of 100 kbps lasting 100 seconds, between 300 and 600 kbps. In Fig. 6, we can see Rebera's predicted bandwidth, overall budget and its

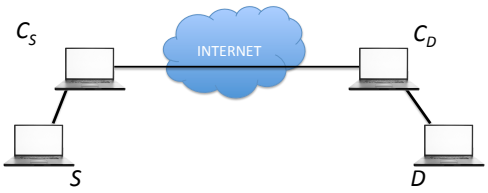

Fig. 5. Illustration of the testbed

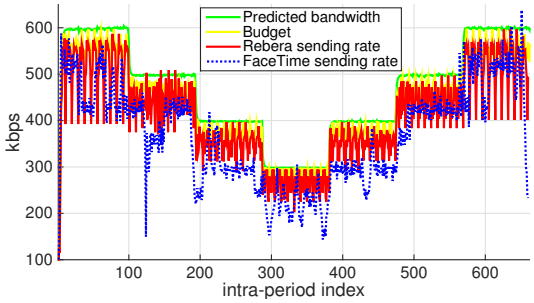

Fig. 6. Bandwidth utilization under staircase bandwidth

sending rate, along with FaceTime's sending rate. On average, the bandwidth utilization of Rebera is 83.32%, while FaceTime achieves a utilization of 77.67%. The periodic negative peaks in Rebera's sending rate, seen clearly when the bandwidth is high, correspond to the intra-period with the smallest rate: for these intra-periods, the budget is higher than the video rate, and Rebera ends up sending all the frames, but still cannot use up most of the budget. As a result, our bandwidth utilization is reduced. Had the encoder adaptively controlled the QP, higher bandwidth utilization would have been expected. But Rebera still achieves a higher average utilization than FaceTime.

2) Evaluation with Cellular Capacity Traces: In this set of experiments, we use cellular bandwidth traces (Fig. 7) to emulate the cellular links. Each experiment lasts for 1000 intra-periods. We first present the results involving a single cellular link along the end-to-end path. We employ traces 2, 4, 5 and 6 as the forward, and traces 1 and 7 for the backward end-to-end available bandwidth. Results are summarized as bandwidth utilization and 95-percentile packet queuing delay tuples in Tables IV and V. Specifically, when trace 1 is used for the backward path, the information fed back to the sender side undergo a longer delay for both Rebera and FaceTime, degrading the performance of both systems. As an example, in Fig. 8, we present Rebera's and FaceTime's sending rates with Trace 2 for forward and Trace 7 for backward available bandwidth, respectively. We can see from Table IV and V, in all experiments, Rebera achieves a higher utilization of the forward available bandwidth with shorter packet delays. Specifically, Rebera provides an average utilization of 1.2015 times more than Face-

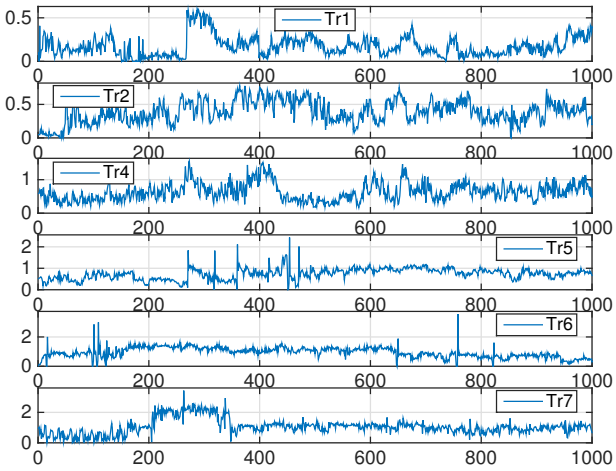

Fig. 7. Traces used in the experiments. Vertical axis is the capacity in Mbps, horizontal axis is the intra-period index. Traces 2, 4, 5 and 6 are used as forward capacities, 1 and 7 are used as backward capacities.

Time, and an average reduction of 113 ms in the 95-percentile packet queueing delays. Furthermore, having a more challenging backward capacity (trace 1 in Table IV) decreases the bandwidth utilization of both Rebera and Facetime. FaceTime's packet delay performance also degrades, however, Rebera is still able to provide similar packet delays. Finally, we have tested both systems assuming both users are connected over different cellular links. We have used trace 2 and 7 for the cellular link capacities on the forward end-to-end path, and assumed the backward path has infinite capacity with only a constant delay of 40 ms, to be able to measure the one-way packet delays. The results can be seen in Table III.

TABLE III
EVALUATION OVER TWO CELLULAR LINKS, USING TRACE 2 AND 7 AS THE FORWARD CAPACITY AND INFINITE BACKWARD CAPACITY

	Rebera	FaceTime
utilization (%)	56.44%,	46.52%
delay (ms)	387	558

TABLE IV
EVALUATION OVER SINGLE CELLULAR LINK, USING TRACE 1 AS THE BACKWARD CAPACITY

Fwd Cap.	Rebera	FaceTime
Tr2	63.44%, 325 ms	55.81%, 428 ms
Tr4	59.67%, 352 ms	31.38%, 631 ms
Tr5	65.11%, 160 ms	60.45%, 364 ms
Tr6	61.85%, 219 ms	56.42%, 298 ms

3) Effect of Confidence Parameter: Next, we investigate the effect of the confidence parameter δ in Section III-C on Rebera. We vary δ from 0.05 up to 0.5, and record the utilization and 95-percentile packet delays in Table VI. Having a larger δ value means the sender can

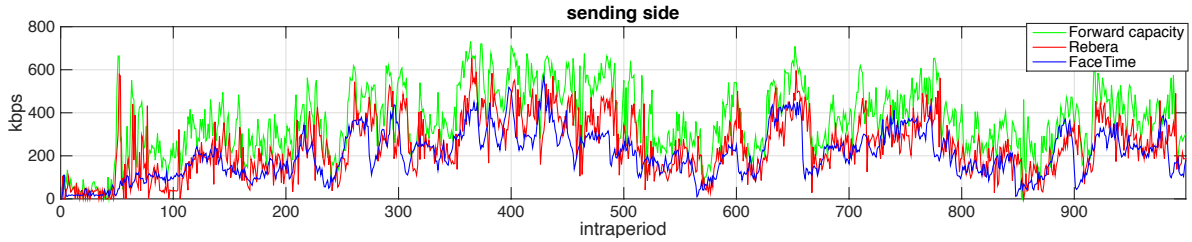

Fig. 8. Video sending rate of Rebera and FaceTime over single cellular link with Trace 2-7 as the forward-backward capacities, respectively

TABLE V
EVALUATION OVER SINGLE CELLULAR LINK, USING TRACE 7 AS THE BACKWARD CAPACITY

Fwd Cap.	Rebera	FaceTime
Tr2	67.18%, 347 ms	56.86%, 407 ms
Tr4	64.89%, 287 ms	59.63%, 322 ms
Tr5	70.97%, 163 ms	67.90%, 257 ms
Tr6	68.08%, 239 ms	62.73%, 288 ms

tolerate larger bandwidth prediction errors, and hence larger packet delays, in exchange for more bandwidth utilization, which could be the case for less interactive video streaming applications.

TABLE VI
EFFECT OF THE CONFIDENCE PARAMETER ON REBERA OVER SINGLE CELLULAR LINK. FORWARD CAPACITY TRACE 2, BACKWARD CAPACITY TRACE 7

δ	0.05	0.1	0.2	0.5
utilization (%)	67.18,	69.63,	74.18	79.77
delay (ms)	347	364	404	468

4) Loss Resilience: To examine the performance of Rebera in presence of packet loss, we employ CellSim to introduce random losses. We have tested Rebera when the packet loss rate is 5% and 10%, and the results are given in Table VII. Although not significantly, the bandwidth utilization drops with the loss rate as expected. However, the delays experienced by the received frames reduce, since there is less backlog in the buffers.

TABLE VII
EFFECT OF PACKET LOSS ON REBERA OVER SINGLE CELLULAR LINK. FORWARD-BACKWARD CAPACITY: TRACES 5-7

loss rate	0	0.05	0.1
utilization (%)	70.97	67.60	64.33
delay (ms)	163	141	129

VI. CONCLUSION

Video calls over cellular links have to adapt to fast-changing network bandwidth and packet delay. In this study, we proposed a new real-time video delivery system, Rebera, designed for cellular networks. Rebera's proactive congestion control module uses the video frames themselves to actively measure the capacity of cellular links, and uses these measurements to make a safe forecast for future capacities, using the well-known adaptive filtering techniques. Through its dynamic frame selection module designed for temporal layered streams, Rebera ensures that its video sending rate never violates the forecast by discarding higher layer frames, thereby preventing self-congestion, and minimizing the packet delays. Our experiments showed that Rebera is indeed able to deliver higher bandwidth utilization and shorter packet delays compared with Apple's FaceTime. In the future work, we are eager to consider UEP among temporal layers, along with incorporating an adaptive video encoder with low-delay rate control into Rebera, which would help to achieve even shorter delays and higher utilization.

REFERENCES

[1] Microsoft, "Skype," http://www.skype.com.
[2] Google, "Hangout," http://www.google.com/+/learnmore/hangouts/.
[3] Apple, "Facetime," http://www.apple.com/mac/facetime/.
[4] X. Zhang, Y. Xu, H. Hu, Y. Liu, Z. Guo, and Y. Wang, "Profiling Skype Video Calls: Rate Control and Video Quality," in *Proceedings of IEEE INFOCOM*, 2012.
[5] H. Alvestrand and S. Holmer, "A google congestion control algorithm for real-time communication on the world wide web," 2012.
[6] K. Winstein, A. Sivaraman, and H. Balakrishnan, "Stochastic forecasts achieve high throughput and low delay over cellular networks," in *USENIX Symposium on Networked Systems Design and Implementation*, 2013, pp. 459–471.
[7] S. Haykin, "Adaptive filter theory," *Prentice Hall*, 2002.
[8] R. Prasad, C. Dovrolis, M. Murray, and KC Claffy, "Bandwidth estimation: metrics, measurement techniques, and tools," *IEEE Network*, vol. 17, no. 6, pp. 27–35, 2003.
[9] Q. He, C. Dovrolis, and M. Ammar, "On the predictability of large transfer tcp throughput," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 4, pp. 145–156, 2005.
[10] B. Farhang-Boroujeny, *Adaptive filters: theory and applications*, John Wiley & Sons, 2013.
[11] A. Arasu and G. S. Manku, "Approximate counts and quantiles over sliding windows," in *Proceedings of ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. ACM, 2004, pp. 286–296.
[12] Y. Liu, Z. G. Li, and Y. C. Soh, "A novel rate control scheme for low delay video communication of h. 264/avc standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 1, pp. 68–78, 2007.
[13] JM Ref. Software, "version 18.6," http://iphome.hhi.de/suehring/tml/.
[14] Xiph.org, "Test Media," https://media.xiph.org/video/derf/.
[15] Visicom Media, "ManyCam," https://manycam.com.