Predicting Quality of Delivery Metrics for Adaptive Video Codec Sessions

Obinna Izima, Ruairí de Fréin, Mark Davis
Technological University Dublin, Ireland

2020 IEEE 9th International Conference on Cloud Networking (CloudNet) | 978-1-7281-9486-8/20/$31.00 ©2020 IEEE | DOI: 10.1109/CLOUDNET51028.2020.9335813

***Abstract*—Predicting video quality will continue to be an active area of research given the dominance of video traffic for years to come. Network service practitioners that are poised to handle the strain on the existing limited bandwidth constraints are better placed to be SLA-compliant. The dynamic and time-varying nature of cloud-hosted services require improved techniques to realize accurate models of the systems. To address this challenge: (1) we propose Codec-aware Network Adaptation Agent (cNAA), an online light-weight data learning engine that achieves accurate and correct predictions of quality of delivery (QoD) metrics, namely jitter for video services. cNAA achieves this prediction accuracy by leveraging the available network information in the face of congestion and adaptive codecs; (2) we highlight the short-comings of some baseline machine learning techniques that fail to capture network dynamics and demonstrate their failure in comparison with cNAA; and finally, (3) we demonstrate the efficacy of cNAA under varying network and codec conditions and provide evidence showing that machine learning approaches that incorporate network dynamics are better placed to realize accurate and correct predictions.**

***Index Terms*—Prediction, Jitter, Adaptive Codecs.**

I. Introduction

Video traffic will continue to dominate the Internet. This is according to the Cisco Visual Networking Index [1] which predicts that annual global IP traffic will reach 4.8 Zettabytes. This trend presents new challenges in guaranteeing the quality of the video content delivered over today's Internet. This is because video delivery deployed over best-effort IP networks are highly dynamic in nature with bandwidth fluctuations and time-varying delays making it a challenge to consistently guarantee the quality of delivery (QoD) of video content over such networks.

The authors of [2] made a timely contribution in their work on video service-level prediction. They considered the problem that service providers should be able to deliver on agreed service-level agreements (SLAs). They used machine learning (ML) techniques to predict client-side metrics for a video streaming service, using variants of Linear Regression, LASSO and Elastic Net.

In this paper, we propose **Codec-aware Network Adaptation Agent, (cNAA)**, a light-weight online learning engine that achieves accurate QoD metric prediction, namely Jitter statistics for a video streaming session between a server and client devices. Our proposed in-network agent achieves accurate estimates of the QoD metric by incorporating network dynamics into the data learning algorithm. We demonstrate that ML-enabled learners that fail to build in the underlying network dynamics fail in their bid to realize accurate QoD predictions.

This paper is organized as follows. In Section II we place our contribution in the context of the related literature. In Section III we introduce the different learning strategies that are evaluated. In Section IV we evaluate the efficacy of each of the approaches and present our conclusions in Section V.

II. Related Works

Predicting the effect of network performance on the perceived video quality is critical, as this determines the success, failure or degradation of the video service. ML techniques have been leveraged in linking the perceived video quality to network- and application-level QoD metrics [3].

Vega et al. [4] investigated the prediction of perceived video quality under QoD impairments. In the work, the authors deployed ML models to assess the quality of video streams in real-time. The features utilized in the study were mainly from the video-related features (such as the bit stream, frame, inter-frame and content) as no network QoD parameters were considered in the work. However, the QoD impairments studied (network delay, jitter, throughput) were treated as independent conditions. Usually, these sort of impairments usually happen together.

In [5], Mushtaq et al. investigated the impact of network QoD metrics, video-related features and viewer features over the perceived video quality. The authors generated a dataset from a controlled network environment where varying degrees of delay, jitter and packet loss are applied to video streams flowing through a network emulator. Network level features such as delay, jitter, packet loss etc., and some application-related features such as resolution, video type, and viewer-related feature such as gender, interest etc, are fed to the ML models and evaluated via subjective testing based on mean opinion score (MOS). However, the work does not describe the parameters of the models studied and also fails to indicate the significance of the selected features. For instance the significance of the viewer-related features was not provided.

The work in [6] characterizes the effect of the system load, i.e. the number of concurrent users accessing a video stream for a cloud-hosted streaming session involving a server and client devices. The authors present evidence that demonstrate the effects of disregarding the load on the system. Building on these results, the authors of [2, 7] contribute adaptive learning methods that have lower computation complexity and yield more accurate predictions than approaches that ignored the presence of the load in the system. In their work, the authors consider various ML linear models and ensemble models and demonstrate that subset selection alone reduce the effectiveness of the prediction. They propose the load-adjusted (LA) learning technique which demonstrate that the learners that achieve accurate and correct predictions are the ones that take the effect of the network dynamic, the load in this case, into account.

Contribution: Figure 1 illustrates the purpose of our proposed in-network prediction agent, cNAA. cNAA, an online ML-enabled agent obtains accurate QoD predictions and parameter estimation by leveraging information about the underlying network dynamics present in the system. The focus of this paper to is to demonstrate that the network conditions hold vital sources of information that can be incorporated into the ML-enabled data learning algorithms to realize accurate predictions of QoD metrics for video services. We incorporate the underlying effect of congestion and the resultant adaptive behaviour of the codecs in our learning engine to demonstrate that learners must incorporate these network dynamics to achieve accurate and correct predictions.

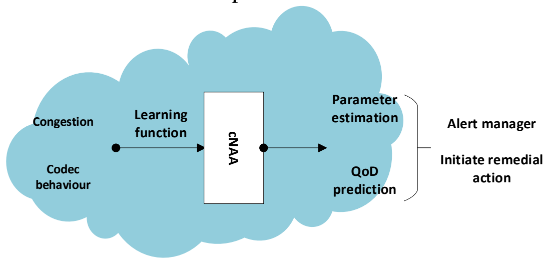

Fig. 1. The role of cNAA agent in the QoD prediction ecosystem. The learning function incorporates network dynamics, namely, congestion and codec behaviour in QoD parameter estimation and prediction. The prediction outputs can be fed into an intermediary network device (e.g. a router) or sent to a network manager to either take remedial or proactive action.

III. Learning Strategies

Network congestion is one primary cause of performance degradation and performance variability for time sensitive applications like videos. As such, congestion control (CC) and prediction is not only integral to networks operations but also required to achieve fairness in resource utilization and minimize packet loss [8]. More so, despite the advancements made in compression technologies, the compression fundamentals barely changed at all. What has changed is that new codecs are equipped with added tools and complexity. Modern video streaming services are codec-agnostic shifting the focus from codecs to how to enable the codec do a better job.

A fundamental architecture for video streaming systems is made up of two major components, a codec to encode the video, and a transport protocol to convey the video content from the source to the destination. Typically, the transport protocol is responsible for figuring out the network capacity by looking at congestion, round trip time (RTT) etc. The codec then utilizes the network capacity to figure out what levels of compression to apply to the video, so that it can adaptively relay the video stream to the destination [9].

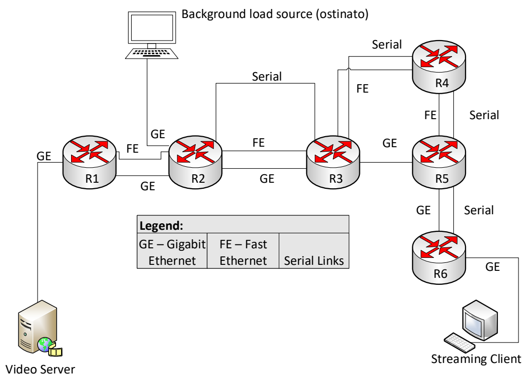

Fig. 2. A video server streams H.264 video over RTP to a streaming client over a topology which consists of 2911 ISR routers. Different link technologies (Gigabit Ethernet, Fast Ethernet and Serial) are used to create a range of different physical transport media, which in turn cause variability in instantaneous jitter values. Ostinato is used to create congestion on different paths through the network.

A. *Experimental Methodology*

Figure 2 illustrates our laboratory network set-up for a video streaming session between a server and a client machine using physical Cisco hardware routers between the streaming server and a client machine. In the network, we connect six (6) routers (R1 - R6, all Cisco 2911 Integrated Services Routers (ISR)) between the server and the client. Both the server and client machines are 64-bit Windows 7 operating system computers. The clocks on the server and the client are synchronized using the Network Time Protocol in order to match observations from both. Samples are drawn from the client machine every second. We set up the streaming session on the server machine with the VLC server using the Real-Time Transport Protocol (RTP). We use the Big Buck Bunny mp4 animation video $\approx$ 10minutes long for our experiments. We activate transcoding of the pre-encoded file to either H.264 or H.265. At this stage, we do not alter any further default VLC encoding parameter leaving the bitrate and frame rate etc., same as the source file.

To evaluate the efficacy of our proposed learning agent in the face of congestion, we instrument routing changes to force the videos streams to switch paths between the paths offering the most bandwidth (i.e. the connections on the GE 1000Mb/s), the FE ports and serial links. Furthermore, we introduce additional congestion in the network through bursts of UDP background traffic ingestion via the GE interface at router, R2 using Ostinato Packet Generator. To capture network metrics for the streaming session, we use Wireshark, a popular network protocol analyzer to collect traces. Our QoD metric of interest is the interarrival jitter statistics of the video packets. In Wireshark, the interarrival jitter is estimated according to [10].

Figure 3 illustrates time-varying periodically falling exponential curves generated as a result of the adaptive behaviour of the codecs in response to congestion in the network. We hypothesize that by learning in the face of network dynamics we can realize accurate and correct predictions. Specifically, our objective is to predict jitter statistics accurately in the face

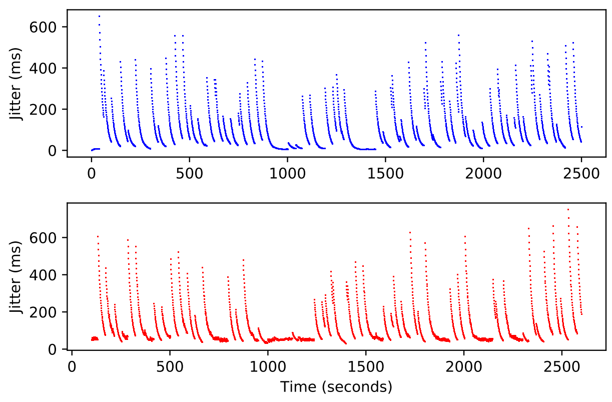

Fig. 3. Rows 1 (R1) and R2 illustrate traces depicting the adaptive behaviour of the H.264 and H.265 codecs respectively in response to network congestion. The maximum, minimum and mean instantaneous jitter values in R1 and R2 are (650ms, 1.17ms, 107ms) and (750ms, 3.91ms, 126ms) respectively. The structured nature of these time series is significant.

of congestion and adaptive codecs.

The slow downward exponential curves shown in Fig. 3 are characterized by time varying network transit times. The network transit times also show periodicity which significantly varies throughout the sessions. Above all, another noticeable feature of the traces are the "spikes" in the network transit times (or periods) which are due to buffering delays or out-of-order packet arrivals [11].

B. Time Series Analysis

A time series $j = (j_0, j_1, \ldots, j_T)$ is an ordered sequence of data points measured in equally spaced time intervals, where $j_t \in \mathbb{R}$ represents an element at time t, $0 \leqslant t \leqslant T$ (T = length of the time series). j_0 represents the beginning of our time series data, $(j_0, j_1, \ldots, j_T)$. Thus, our task is to estimate $\hat{j}_t$.

We start our analysis with the linear regression (LR) model. It models the relationship between the current jitter value, j_t, and previous jitter values.

$$\hat{j}_t = \sum_{i}^{P} w_i j_{t-1} + b. \tag{1}$$

The LR model consists of linear functions of P inputs parameterized in terms of P coefficients, the weights, w_i, and an intercept term, b. The LR loss function quantifies how good the linear fit realized by the model is. This is a function $\mathcal{L}(\hat{j}_t, j)$ which indicates how far off the prediction $\hat{j}$ is from the actual value j [12]. In LR, we use the squared error, defined as:

$$\mathcal{L}(\hat{j}_t, j) = \frac{1}{2}(\hat{j}_t - j)^2. \tag{2}$$

A commonly used technique for time series prediction given its beginning is the autoregressive (AR) model, moving average (MA) model, autoregressive-moving average (ARMA) model and the exponential weighted moving average (EWMA) models [13].

Given a time series and a fixed subset size n, the simple moving average (or rolling average) model estimates are realized by taking the average of the initial fixed subset of the time series. Then, the subset is modified by shifting forward in time over the series. That is, excluding the initial set of samples and including the new values in the subset. Some limiting factors of the SMA are that its estimates lags by the size of the window and extreme historical values may skew the performance. The EWMA helps reduce the lag effect from SMA and puts more weight on recent data points (by applying more weight to the recent values).

The AR model is actually a linear regression whereby a value from the time series data is regressed on prior values from the same series. A common AR model extension uses a moving average of (MA) giving rise to the autoregressive-moving average (ARMA) model. Similar to the AR model, ARMA(p,q) is a linear regression of the instantaneous value of the time series against one or more previous values of the series and one or previous noise terms. The values of p indicates the order of the AR part of the model and the q is the order of the MA part of the model.

C. The Proposed Method

Figure 4 shows an extract from the data shown in Row 1, Fig. 3. Clearly, the curves are of the form of an exponentially decaying data. Thus, we assume the unit exponential decay function:

$$J(t) = J_0 e^{\lambda t}, \tag{3}$$

where $J(t)$ represents the jitter value at time, t, J_0 is the initial jitter value represented by the peak value or height of the curve, and λ is the decay constant.

In addition, the curves are separated by time-varying periods, P_T.

$$J(t) = J(t + P_T). \tag{4}$$

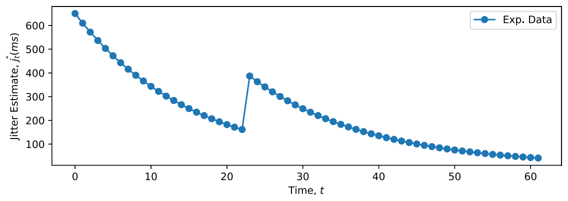

Fig. 4. An extract of the data shown in Row 1, Fig. 3 showing the exponential behavior of the curves.

Consequently, our goal is to realize a model that estimates all three important parameters, namely, J_0, λ and P_T. To fit an individual curve, Equation 3 can be expressed as a linear model in the form of Equation 1 by taking the natural log of both sides:

$$\ln J_t = \ln J_0 + \lambda t. \tag{5}$$

To solve for the parameters, J_0 and λ, we can apply the LR model loss function in Equation 2 and obtain a solution using the LR normal or closed form equation. However, we need to realize a model that can reliably estimate all parameters including the period, P_T as any solver that fails to capture all three parameters would produce wrong estimates.

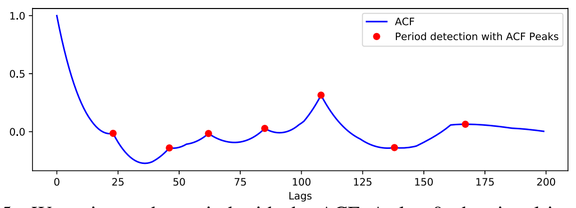

Fig. 5. We estimate the period with the ACF. At lag 0, the signal is exactly itself. If periodicity exists in the signal, we see a significant increase in the correlation. Peaks (red circles) signifying the indices of the periods are discovered by searching for values which are surrounded by lower values for maxima.

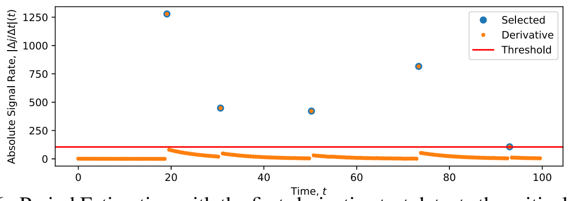

Fig. 6. Period Estimation with the first-derivative test detects the critical points in the signal signifying the indices at which the signal changes direction. The threshold sets the boundary for all detected derivatives or critical points. This can also be automated with a peak detection algorithm which can search for maximum points surrounded by lower values.

1) Period Estimation with Auto-correlation Function: A common tool for detecting the period in a signal is with the auto-correlation function (ACF). Typically, the ACF sequence of a periodic signal exhibits the same cyclic characteristics as signal itself. Thus, the ACF can assist in determining the presence of cycles as well as their duration [14]. Using the ACF, we shift the signal with a time lag and compute the correlation with the original signal. After a certain time lag, we see the correlation exhibits a significant increase as shown in Fig. 5. The peaks (shown as red circles) in the ACF figure indicate the periodic signal and its corresponding time lags.

2) Period Estimation with First-Derivative Estimates: The first-derivative criterion examines a function's monotonic properties (where the function is either increasing or decreasing) and focuses on a critical point in its domain [15]. The test enables us detect when the curves drastically go up or down which reveals the periods between the curves. We compute the first-derivative estimates for our time series data shown in Fig. 6 by taking the pairwise differences of the signal with respect to time to detect the critical point.

The criterion requires an extra parameter (threshold) that must be tuned accordingly to the signal specifications. Mainly the criterion threshold will be affected by the nature and the power of the noise on your data and the gap magnitudes between two curves. The usage of this methodology depends on the ability to detect curves gap in presence of noise. It will break when the noise power has the same magnitude of the gap we want to detect. In this series, we have set the threshold to 105 $[SignalUnits/TimeUnits]$. This threshold can also be computed using any peak detection algorithm.

The use of the first-derivative test is a reasonable one to use in this scenario as the curves in our time series are smooth as we believe the video server is adapting the video play-out in a deterministic way. Armed with either technique, we can estimate the periods, P_T of the curves in the traces and fit a model to estimate our parameters and make predictions.

D. Model Fitting

In this section, we describe two model fitting procedures for our proposed agent. We compare the performance of our approach against some baseline ML models LR, ARMA, SMA and EWMA which are described in III-B.

1) Split Data and Model Fit to Independent Curves: We begin by extracting the first 100 timestamps in our time series data. The first step is to detect the periods of each curve in the dataset. We estimate the periods using the first derivative test. The next step is to divide the data using the period estimates and fit a model to the individual curves in the dataset. To fit a model to the subsets of the data separated by the time-varying periods, we use the models described in Equations 3 & 4, Section III-C to fit the independent curves. The time origin shift is handled by the splitting methodology. The predictions are shown in Fig. 7.

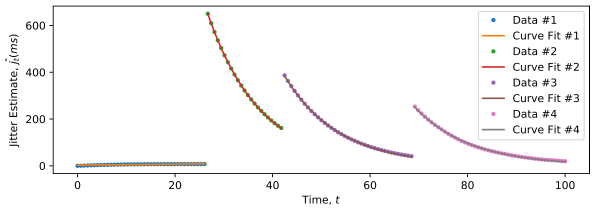

Fig. 7. cNAA Approach 1 - Accuracy of the cNAA jitter predictions. The predictions obtained for the independent curves are overlaid on the actual jitter values.

2) Parameter Estimation from Historical Data: Using the technique described in Section III-D1, we realize accurate fit to the curves by first detecting the periods between curves. However, if we were to predict timestamps between 60 - 65 in 7, we would do that based on timestamps from 41 - 60. In this section, we propose model parameter estimation and prediction based on the historical values of the data.

To do this, we overlay some curves on the time axis and obtain a model by finding the best fit through all data points. Specifically, with this approach, the parameter estimates could be used to predict what the jitter estimates could be from 60 - 65. This is a reasonable assumption because the timestamps from 0 - 10, will contain relevant information to predict what happens from timestamps 11 - 23. Similarly, timestamps 24 - 30, will contain relevant information to predict 35 - 40. In Fig. 8, we illustrate a model fit obtained for 3 curves overlaid on the time axis. The thick blue line is the model fit obtained which contains the relevant information that estimate the jitter values for all data.

IV. Evaluation and Analysis

Our proposed algorithm, alongside the baseline models described in Section III-B are evaluated using the same dataset and under same conditions as described in III-D. The model performance are evaluated using the root mean squared error (RMSE) and the mean absolute error (MAE).

The LR model fails to capture the network dynamic, the time varying periodic data and the data shape. The LR records

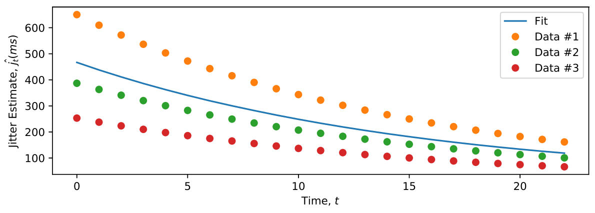

Fig. 8. cNAA Approach 2 - Model fit realized from parameter estimation from historical data. This approach leverages historical data in model parameter estimation and predictions.

the worst performance of all models compared with our proposed approach with an RMSE and MAE of 119.66 and 91.92 respectively.

To determine the (p,q) order of the ARMA model to apply on our time series data, we first utilize an automated grid serach using Python's forecast tool, Pmdarima. The tool suggests an ARMA (2,1) model. We confirm these model suggestions by using the partial autocorrelation function (PACF) and the ACF. The PACF can reveal the recommended AR(p) order and an ACF plot can reveal the MA(q) order [16]. Both functions confirm an ARMA model(2,1).

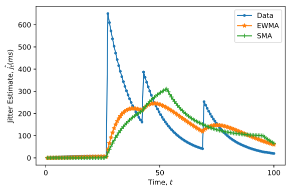

Fig. 9. Prediction performance of the baseline models (EWMA and SMA) compared with the actual data. The EWMA performs better than the SMA. However, both algorithms when compared with cNAA offer very poor predictions.

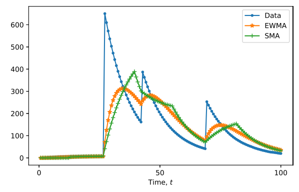

Fig. 10. We obtain improved prediction performance in both the EWMA and SMA by building in network dynamics in the models. There is $\approx$ 19% and 15% improvements for EWMA and SMA respectively.

For both the SMA and the EWMA, we start by setting the sliding windows with the period estimates computed from the data. The resultant predictions are demonstrated in Fig. 9. We then halve the periods and re-run both algorithms. Here, we attempt to incorporate historical data in our model estimates. Consider timestamps 0-39 in the first curve shown in Fig. 7, halving the window would enable us incorporate historical data from 0-19 in estimating 20-39 timestamps. The resultant predictions are shown in Fig. 10.

TABLE I
CODEC-AWARE NETWORK ADAPTATION AGENT (cNAA) VERSUS BASELINE MODELS; COMPARED WITH THE BEST PERFORMING BASELINE MODEL, EWMA, cNAA TECHNIQUES OFFER $\approx$ 95.8% AND 93.8% PERFORMANCE IMPROVEMENTS IN TERMS OF RMSE AND MAE RESPECTIVELY.

Method	RMSE	MAE
cNAA	**4.82**	**4.07**
EWMA	115.84	65.50
SMA	138.88	81.24
LR	119.66	91.92
ARMA(2,1)	119.16	89.76

A. Results and Discussion

Table I compares the performance of our proposed cNAA technique against the baseline models. The RMSE and MAE is $\approx$ 95.8% and 93.8% better respectively over the best performing baseline model, EWMA. The best performing baseline model, EWMA doesn't even offer any close competition in terms of both metrics compared to cNAA. The ARMA model with 119.16 and 89.76 as RMSE and MAE respectively is $\approx$ 114.34 and 84.94 packets less accurate than the cNAA model. The SMA records the worst RMSE with a difference of about 134 packets. The significant performance gain of the cNAA is obtained by incorporating the vital network dynamics. We attempt to build in some of the dynamics in the SMA and EWMA as demonstrated in Fig. 10 by halving the periods in order to incorporate historical information within the sliding windows of both algorithms.

TABLE II
SMA AND EWMA OFFER BETTER PREDICTIONS BY BUILDING SOME NETWORK DYNAMICS INTO THE MODELS. WE RECORD $\approx$ 19% AND 15% IMPROVEMENTS IN PREDICTIONS BY BUILDING IN SOME NETWORK INFORMATION.

Method	RMSE	MAE
EWMA	93.71	49.48
SMA	113.16	62.41

Evidently, both algorithms computations are by far off especially around the critical points. However, halving the periods does improve the predictions by both algorithms as demonstrated in Table II. We are able to realize $\approx$ 19% improvement in the EWMA prediction performance by incorporating some dynamics in the model. Similarly, for the SMA model, the improvement in prediction performance is $\approx$ 15%.

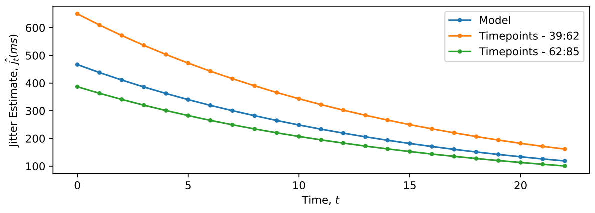

Fig. 11. Accuracy of model fit obtained by estimating the parameters from historical data. The jitter estimates obtained closely approximates the data.

In Fig. 11, we demonstrate the efficacy of our proposed described in Section III-D2. We have taken some timestamps from the second and third curves shown (indices 39-62 & 62-101) in Fig. 7 obtained a model for these data points by estimating the parameters from historical data. We show that we can approximate a fit for these data points.

We have given evidence that the cNAA approach offers accurate and correct QoD metric prediction, namely jitter in the face of congestion and adaptive codecs. We now consider the effect of varying the congestion levels in the network on the cNAA proposed method.

We evaluate our proposed method with the traces shown in Fig. 12. One trace is a 2-state dataset of low and high congestion levels. The other trace, is a 3-state dataset with an intermediate congestion level in addition to low and high congestion. The 2-state periods are at indices 0, 14, 27, 40, 52 and so on. The 3-state dataset periods are at indices 0, 13, 50, 56, 67 etc. Both start off at somewhat similar rate before the jump in the 3-state trace. We also observe that the periods in both traces are shorter with more spikes.

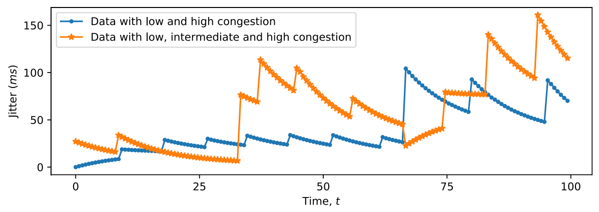

Fig. 12. Data with varying levels of congestion obtained by varying the link technologies i.e. the GE, FE and Serial links which in turn generate variability of instantaneous jitter. By altering the paths across these links, we generate varying degrees of congestion levels.

Next, we vary the bitrates of the codecs to evaluate the performance of cNAA. We generate two datasets by setting the bitrate to 56 kilobits per second (Kb/s) in one trace and 1000 kb/s in the other trace. The frame rates are set to 24 frames per second. In the 1000 kb/s trace, the audio codec settings are same as the source file whereas we set the audio codec bitrate at 24 kb/s in the other trace. The traces are shown in Fig. 13.

The 1000 kb/s trace starts off slowly with a lot of small spikes and very short periods. Generally, there are obviously a lot of perturbations in the system. The 56 kb/s trace exhibits the same trend we have seen before. The data starts slowly and then spikes and maintains some periodicity in its network transit times between the curves. We proceed to evaluate cNAA with these traces.

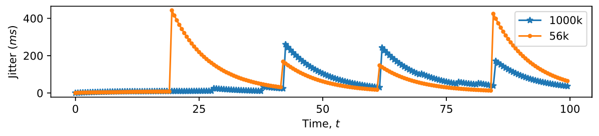

Fig. 13. Data generated by varying the encoding parameters of the codec. We set the video bitrate to 56 kb/s and 1000 kb/s to generate varying levels of instantaneous jitter.

We obtain an accurate model for the 56 kb/s data. The jitter estimates generally follow the actual jitter values very well. However, there are a few misses in the 1000 kb/s trace. The slow and almost negligible periodicity at the beginning of the trace is missed by our predictions. Also, timestamps between 65 - 85 show that there are about 3-4 curves almost overlapping on each other. The estimates generated here attempts to approximate all data points as one rather than as independent curves as expected.

Based on our evaluation and analysis, the model of choice for predicting network QoD metric, jitter is the cNAA. The cNAA boasts the ability to overcome the limitations of the baseline models by building in network dynamics in its learning process. Our results also demonstrate that cNAA offers accurate predictions in the face of changing congestion and bitrates levels. Table III lists the performance of cNAA under varying congestion levels and bitrates. The worst performance of the cNAA is with the 1000 kb/s model with high system excitation. While the RMSE and MAE metrics are highest in this case, the cNAA predictions are generally accurate and correct as well. In a novel approach, we evaluated how the cNAA technique could be applied to predict network QoD metrics for video services. We note that the performance of the agent offers accurate estimates of the QoD metric, jitter without requiring any additional complexity in terms of computational power. In future work, we will investigate how we can achieve network state acquisition [17] and monitoring under time-varying network conditions with our model parameters.

TABLE III
CODEC-AWARE NETWORK ADAPTATION AGENT (CNAA) PREDICTION PERFORMANCE IN VARYING CONGESTION LEVELS AND BITRATES. CNAA OFFERS ACCURATE AND CORRECT PREDICTIONS IN ALL CASES. THE WORST PERFORMANCE OCCURS IN THE 1000 KB/S TRACE DUE TO HIGH SYSTEM PERTURBATIONS.

Model	RMSE	MAE
2-state	7.05	6.07
3-state	5.00	4.29
56 kb/s	4.61	3.90
1000 kb/s	14.93	12.31

V. CONCLUSION

In this paper, we propose a Codec-aware Network Adaptation Agent (cNAA), an in-network agent that achieves accurate and correct predictions of QoD metrics, jitter in the face of congestion and adaptive codecs. cNAA's performance was compared with some baseline ML techniques which offered very poor predictions. The performance gain of cNAA was achieved by leveraging the information in the network. We demonstrate cNAA's prediction accuracy under varying network and codec conditions. We also demonstrated with some baseline models that by incorporating network dynamics, we realize improvements in prediction accuracy. cNAA as an online learning engine could be deployed in a network to assist network managers predict the network performance or for network monitoring purposes. The model estimates could also be passed off to some network device, for instance, a router to effect a change in routes etc.

As part of our future work, we propose to extend the functionalities of the cNAA towards network monitoring. We will look towards implementing a network traffic classifier, whose feature space the cNAA model estimates.

Furthermore, the results we present show that the cNAA technique reacts in a deterministic way when the video delivery model uses an adaptive codec. We will investigate scenarios with multiple flows with the fundamental architecture remaining unchanged. In such scenarios, we hypothesize that an algorithm which builds in the relevant network information would offer better prediction performance.

Given the increased dynamicity of modern service delivery, this lightweight, data-learning engine offers a lot of promise to practitioners as they show that off-the-shelves techniques may be improved without any additional requirement for computational power or data.

Acknowledgements

This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under the Grant Number 15/SIRG/3459.

References

[1] Cisco Systems 2018, "Cisco Visual Networking Index: 2017-2022 White Paper," *Cisco*, 2018.

[2] O. Izima, R. de Fréin, and M. Davis, "Video Quality Prediction Under Time-Varying Loads," in *IEEE CloudCom*, 2018, pp. 129–132.

[3] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, "A Comprehensive Survey on Machine Learning for Networking: Evolution, Applications and Research Opportunities," *J. Int. Ser. App.*, vol. 9, no. 16, June 2018.

[4] M. Torres Vega, D. C. Mocanu, and A. Liotta, "Unsupervised Deep Learning for Real-Time Assessment of Video Streaming Services," *Multi. Tools Appl.*, vol. 76, no. 21, p. 22303–22327, Nov. 2017.

[5] M. S. Mushtaq, B. Augustin, and A. Mellouk, "Empirical study based on machine learning approach to assess the QoS/QoE correlation," in *European Conf. on Net. and Opt. Comm.*, 2012, pp. 1–7.

[6] R. de Fréin, "Effect of System Load on Video Service Metrics," in *ISSC*, 2015, pp. 1–6.

[7] O. Izima, R. de Fréin, and M. Davis, "Evaluating Load Adjusted Learning Strategies for Client Service Levels Prediction from Cloud-hosted Video Servers," in *AICS*, vol. 2259, 2018, pp. 198–209.

[8] A. Bhatele, A. R. Titus, J. J. Thiagarajan, N. Jain, T. Gamblin, P. Bremer, M. Schulz, and L. V. Kale, "Identifying the culprits behind network congestion," in *IEEE IPDPS*, 2015, pp. 113–122.

[9] 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, pp. 267–282.

[10] "RFC 3550 - RTP: A Transport Protocol for Real-Time Applications," https://tools.ietf.org/rfcmarkup?rfc=3550draft=url=section-6.4.1, (Accessed on 09/19/2020).

[11] Collin, Perkings, *RTP: Audio and Video for the Internet*. Publisher: Addison Wesley, pp. 31-32, 2003.

[12] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Publisher: Springer, pp. 43-56, 2001.

[13] G. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*, 5th ed. Wiley, 2016.

[14] Li Hui, Bei-Qian Dai, and Lu Wei, "A Pitch Detection Algorithm Based on AMDF and ACF," in *IEEE ICASSP*, vol. 1, 2006.

[15] "A Pragmatic Introduction to Signal Processing with Applications in Scientific Measurement," https://terpconnect.umd.edu/ toh/spectrum/TOC.html, online; accessed 12 September, 2020.

[16] Duke University Statistical Forecasting, "Identifying the orders of AR and MA terms in an ARIMA model," https://people.duke.edu/ rnau/411arim3.htm, online; accessed 12 September, 2020.

[17] R. de Fréin, "State Acquisition in Computer Networks," in *IFIP Network*, 2018, pp. 1–9.