(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
21 September 2017 (21.09.2017)

WIPO | PCT

(10) International Publication Number
WO 2017/161122 A1

(51) International Patent Classification:
H04N 19/102 (2014.01) *H04N 7/24* (2011.01)
H04N 19/50 (2014.01)

(21) International Application Number:
PCT/US2017/022715

(22) International Filing Date:
16 March 2017 (16.03.2017)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/309,069 16 March 2016 (16.03.2016) US

(71) Applicant: **UNIVERSITY OF FLORIDA RESEARCH FOUNDATION, INCORPORATED** [US/US]; 223 Grinter Hall, Gainesville, FL 32611 (US).

(72) Inventors: **WU, Dapeng, Oliver**; 3584 SW 30th Way, Apt. 132, Gainesville, FL 32608-2772 (US). **SUN, Kairan**; 3700 SW 27th St., Apt. 1D-203B, Gainesville, FL 32608 (US).

(74) Agent: **WALSH, Edmund, J.**; Wolf, Greenfield & Sacks, P.C., 600 Atlantic Avenue, Boston, MA 02210-2206 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report (Art. 21(3))*

(54) Title: SYSTEM FOR LIVE VIDEO STREAMING USING DELAY-AWARE FOUNTAIN CODES

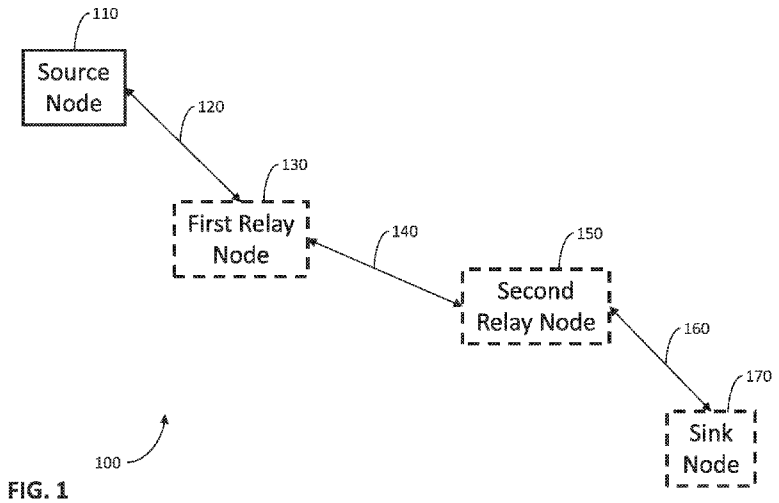

FIG. 1

(57) Abstract: A network system for increasing data throughput and decreasing transmission delay along a data link from a source node to a sink node via a relay node. The network system may comprise: a first node configured to: predict at least one future video encoding characteristic of a video source over a finite number of future frames, encode a plurality of video data packets using rateless coding based on the predicted at least one future video encoding characteristic by adaptively adjusting at least one parameter of the rateless coding based on the predicted at least one future video encoding characteristic, and transmit the plurality of video data packets, wherein the video source is a video source of the plurality of video data packets.

WO 2017/161122 A1

SYSTEM FOR LIVE VIDEO STREAMING USING DELAY-AWARE FOUNTAIN CODES

RELATED APPLICATIONS

This application claims priority to and the benefit of U.S. Provisional Patent Application Number 62/309,069, entitled "SYSTEM FOR VIDEO STREAMING USING DELAY-AWARE FOUNTAIN CODES," filed March 16, 2016. The entire contents of the foregoing are hereby incorporated herein by reference.

BACKGROUND

Over the recent decade, the video-based services have seen explosive growth, especially after the prevalence of smart mobile devices. People are more willing to use YouTube, FaceTime, Netflix, etc., on the move. Thanks to the evolution of communication technologies, such as 3G/4G, LTE, WiFi, etc., the service quality over wireless networks has improved dramatically. However, with the even faster growing video qualities (720p, 1080p, 4K) and the stochastic nature of wireless channels, new challenges are constantly raised for both academia and industry.

In order to deal with the stochastic loss in a wireless network the Forward Error Correction (FEC) codes were developed. One important class of FEC codes are fountain codes [1], such as Luby transform (LT) code [2] and Raptor code [3]. With fountain codes, encoded packets are generated from a given set of native packets, such that the original file can ideally be recovered as long as the number of the received packets is equal to or only slightly larger than the number of native packets. For example, given a file consisting of a sufficiently large number of native packets k, with the latest generation of fountain codes, the RaptorQ codes [4], the chance of decoding failure is less than 1% upon receiving k packets, and less than one in a million upon receiving $k+2$ packets [5]. A major advantage of fountain codes is ratelessness, i.e., the coding rate can automatically adapt to the wireless condition without demanding acknowledgments (ACKs) or retransmissions.

However, traditional fountain codes are designed for file transfer rather than video streaming. Under the current fountain codes paradigm, the users may not start to watch it until the whole video file is successfully decoded.

In order to accommodate fountain codes to video streaming, several window-based

fountain codes have been proposed. In terms of window structure, (i) block coding [6], [7] divides the video file into fixed-length data blocks and separately sends them using fountain codes; (ii) sliding window [8]-[10] segments the video into overlapping windows, encodes them with fountain codes, and decodes them jointly; (iii) expanding window [11], [12] expands the previous window to include new packets. In the above three schemes, the block size is fixed in block coding, virtually extended in sliding window, and keeps growing in expanding window.

SUMMARY

A network system for increasing data throughput and decreasing transmission delay along a data link from a source node to a sink node via a relay node. The network system may comprise a first node configured to: predict at least one future video encoding characteristic of a video source over a finite number of future frames, encode a plurality of video data packets using rateless coding based on the predicted at least one future video encoding characteristic by adaptively adjusting at least one parameter of the rateless coding based on the predicted at least one future video encoding characteristic, and transmit the plurality of video data packets, wherein the video source is a video source of the plurality of video data packets.

At least one computer-readable storage medium encoded with executable instructions that, when executed by at least one processor, cause the at least one processor to perform a method for transmitting a video stream over a data link from a source node to a sink node via a relay node. The method may comprise: predicting at least one future video encoding characteristic of a video source over a finite number of future frames; encoding a plurality of video data packets using rateless coding based on the predicted at least one future video encoding characteristic by adaptively adjusting at least one parameter of the rateless coding based on the predicted at least one future video encoding characteristic; and transmitting the plurality of video data packets, wherein the video source is a video source of the plurality of video data packets.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram illustrating an exemplary source node, relay nodes, and a sink (or destination) node of a network in which some embodiments of the application may be implemented.

FIG. 2 is a flowchart of an exemplary method of increasing data throughput and decreasing transmission delay along a data link from a source node to a sink node according to some embodiments.

FIG. 3 is a flowchart of an additional exemplary method of increasing data throughput and decreasing transmission delay along a data link from a source node to a sink node according to some embodiments.

FIG. 4 is a flowchart of an exemplary fountain code process according to some embodiments.

FIG. 5 is a set of charts illustrating a comparison between bit rates and sampling distributions using (a) foreman, (b) DAF-O, (c) DAF-M, and (d) DAF according to some embodiments.

FIG. 6 is a chart illustrating exemplary comparisons of DAF-L, DAF-O, DAF-M, and DAF according to some embodiments.

FIG. 7 is a chart illustrating exemplary comparisons of DAF-L, DAF-O, DAF-M, DAF, and Block coding according to some embodiments.

FIG. 8 is a set of charts illustrating exemplary comparisons of DAF, DAF-L, DAF-O, and DAF-M according to some embodiments.

FIG. 9 is a chart illustrating a comparison of exemplary fountain code schemes according to some embodiments.

FIG. 10 is a diagram illustrating a computer system on which some embodiments of the invention may be implemented.

DETAILED DESCRIPTION

The inventors have recognized and appreciated that window-based fountain codes may introduce extra play delay due to window size and decoding time. Moreover, window-based fountain codes for video streaming may trade delay for higher decoding performance. The reason is that larger block size may introduce longer delay, but may also provide higher coding gain at the same time [13].

In fact, different video streaming applications may have different levels of delay tolerance. Some applications may be very delay-sensitive, such as live video chat, cloud gaming, etc., all of which are characterized by bi-directional communication [14]. Some uni-directional applications, on the other hand, have a loose tolerance on end-to-end delay, such as TV broadcasting, Internet live streams. The others, such as the streaming of

pre-recorded content, are the least sensitive to delay.

The inventors have recognized and appreciated that Delay-Aware Fountain (DAF) codes may be suitable for video streaming, as proposed in [10]. Different from the other sliding window fountain codes schemes, DAF codes do not treat the sliding windows as homogeneous. By adaptively selecting the window length and adjusting the sampling pattern according to the ongoing video bit rate, DAF codes may deliver significantly higher video decoding ratio than existing schemes.

The inventors have recognized and appreciated, however, that the high-complexity global optimization process of DAF codes may prevent its applications in delay-sensitive video streaming (e.g., live streaming). As a result, the inventors have recognized and appreciated that reduceing complexity while maintaining relatively high coding performance may improve the video watching experience for delay-sensitive video streaming. The inventors have recognized and appreciated that using Model Predictive Control (MPC) may be most appropriate.

MPC has developed considerably over the recent years. As an advanced method of process control, it has been successfully used in the control process of chemical plants, oil refineries, power plants, etc. [15]. Besides, there are also some successful applications in academia, ranging from energy management [16] to automatic vehicle control [17]. Despite the popularity in automatic control community, MPC has never been used in the domain of multimedia communication.

The inventors have recognized and appreciated that integrating MPC with DAF codes may improve video streaming performance, while maintaining low complexity. In MPC, actions may be optimized according to a state sequence over a horizon. Based on that idea, two schemes are discussed herein, i.e., DAF-M and DAF-O.

The inventors have recognized and appreciated that DAF-M may improve upon adjusting the sampling pattern for fountain codes according to the ongoing video bit rate, so as to deliver a consistent video watching experience. The computational complexity may be affordable for live streaming because the objective function may be minimized over a limit-sized horizon, which may not grow with the video length.

The inventors have recognized and appreciated that the online algorithm of DAF-O may be implemented based on the prediction of future bit rate. The inventors have recognized and appreciated that DAF-O may provide a decoding ratio that is significantly higher than existing online video streaming schemes.

Implementation of the System

FIG. 1 is a diagram illustrating a system 100 that may employ techniques for increasing data throughput and decreasing transmission delay from a source node to a sink node via a relay node as described herein. In the example of FIG. 1, a source node 110 (which may be referred to as first node) may encode data packets for transmission. In some embodiments, the source node 110 may encode the data packets using fountain coding (as illustrated at stage 210 of FIG. 2). However, any suitable coding, including rateless coding, may be used to encode the data packets. The source node 110 may also transmit the data packets to a first relay node 130 via connection 120 (as illustrated at stage 220 of FIG. 2), which may be a wireless connection. However, any suitable connection or communication technology may be used to communicate among the nodes.

The first relay node 130 may receive at least one of the data packets from the source node 110. In addition, the first relay node 130 may relay or transmit the data packets to a second relay node 150 via connection 140, which may be a wireless connection. The second relay node 150 may receive at least one of the data packets from the first relay node 130. In addition, the second relay node 150 may relay or transmit the data packets to a sink node 170 via connection 160, which may be a wireless connection.

In some embodiments, source node 110 may be a server, such as a streaming video server and/or a live streaming video server streaming live video content. Alternatively or additionally, source node 110 may include a network ingress point, such as a gateway to a wireless network (e.g., a base station in a wireless network).

Additionally, sink node 170 may be a client, such as a video receiver and/or playback device. For example, sink node 170 may be a client of the server referred to as source node 110. Sink node 170 may be a wireless device, such as a smartphone, tablet, laptop computer, or desktop computer.

Alternatively or additionally, relay nodes, such as first relay node 130 and/or second relay node 150, may include network routers and/or network switches. In some embodiments, relay nodes, such as first relay node 130 and/or second relay node 150, may include hubs, and/or any other suitable components. Alternatively or additionally, relay nodes may include other cell transceivers in a cellular network, such as a 5G network.

In some embodiments, the first relay node 130 and/or the second relay node 150 may regenerate, re-encode, and relay the data packets conditionally, based on the quantity of the

data packets received at the given relay node. For example, the first relay node 130 and/or the second relay node 150 may receive a subset of the data packets, and based on the subset of the data packets, the first relay node 130 and/or the second relay node 150 may regenerate the data packets, re-encode the regenerated data packets, and transmit the regenerated, re-encoded data packets.

The sink node 170 may receive one or more data packets from the second relay node 150. If the sink node 170 has received a sufficient quantity of the data packets, the sink node 170 may regenerate and decode the data packets.

FIG. 1 shows only two relay nodes, the first relay node 130 and the second relay node 150. This number of relay nodes is shown for simplicity of illustration. It should be appreciated that a network system may have many more nodes and relay nodes.

In some embodiments, source node 110 may predict at least one future video encoding characteristic of a video source over a finite number of future frames (as illustrated at stage 210 of FIG. 2). A video encoding characteristic may include any characteristic or property of how a video source is encoded, such as frame rate, number of frames in a group of pictures, video bit rate, and so on. Additionally, source node 110 may encode a plurality of video data packets using rateless coding (as described above) based on the predicted at least one future video encoding characteristic of the video source (as illustrated at stage 220 of FIG. 2) by adaptively adjusting at least one parameter of the rateless coding based on the predicted at least one future video encoding characteristic (as illustrated at stage 225 of FIG. 2). The video source may be a video source of the plurality of video data packets. Additionally, source node 110 may transmit the video data packets. For example, source node 110 may transmit the video data packets over a data link to a second node, which may be first relay node 130, second relay node 150, and/or sink node 170 (as illustrated at stage 220 of FIG. 2). For example, the second node may be a sink node configured to receive one or more of the plurality of video data packets from the streaming video server via at least one relay node, or at least one relay node configured to receive at least one of the plurality of video data packets from the streaming video server. In some embodiments, the data link is at least partially wireless.

In some embodiments, the rateless coding may comprise fountain coding. Alternatively or additionally, the video source may include video data, such as a video file. Alternatively or additionally, at least one video data packet of the plurality of video data packets comprises at least 100 bits, although any number of bits may be used.

In some embodiments, the plurality of overlapping sliding data windows may be non-homogeneous, as discussed herein. For example, the plurality of overlapping sliding data windows may collectively have more than one length and/or more than one sampling distribution.

In some embodiments, source node 110 may obtain at least one video encoding characteristic by preprocessing the video source, and source node 110 may use the at least one video encoding characteristic as well as the predicted at least one future video encoding characteristic. For example, source node 110 may predict the at least one future video encoding characteristic based on at least one video encoding characteristic of the video source.

In some embodiments, adaptively adjusting the at least one parameter of the rateless coding may comprise adaptively selecting a first length of a first data window and a second length of a second data window of a plurality of overlapping sliding data windows (as illustrated at stage 226 of FIG. 3). Additionally, the selecting may use model predictive control based on the predicted at least one future video encoding characteristic.

In some embodiments, the selecting may be based on a number of bits in frames of the video source and/or on a first number of frames in the video source. Additionally, the first length may comprise a second number of frames that the first data window can accommodate, and the second length may comprise a third number of frames that the second data window can accommodate.

In some embodiments, source node 110 may store the first length of the first data window in a header of a first packet of the first data window and may store the second length of the second data window in a header of a second packet of the second data window. In addition, source node 110 may store the first sampling distribution in the header of the first packet of the first data window as a first slope factor and may store the second sampling distribution in the header of the second packet of the second data window as a second slope factor.

In some embodiments, adaptively adjusting the at least one parameter of the rateless coding may comprise adaptively adjusting a first sampling distribution for the first data window and a second sampling distribution for the second data window (as illustrated at stage 227 of FIG. 3). Additionally, the at least one future video encoding characteristic may include a future video bit rate of the video source. For example, the adjusting may be based on a future video bit rate of the video source. In addition, the future video bit rate may be variable.

In some embodiments, source node 110 may segment data from the frames of the video

source into the plurality of video data packets based on at least the first length and the second length (as illustrated at stage 228 of FIG. 3).

In some embodiments, the at least one video encoding characteristic may comprise a past frame rate of the video source, a number of frames in a group of pictures of the video source, and/or a past video bit rate of the video source. Additionally, the past video bit rate may be variable.

Related Work

First of all, the important notations that will be used in this article are listed in Table 1.

Table 1: Definitions of the notations.

Notation	Unit	Definition
T	frame	Video length. Total number of frames in the video sequence.
R	byte/second	Data rate.
C	N/A	Code rate.
W	frame	Sliding window size.
T_{Delay}	frame	Tolerable delay from the video being generated to it being played.
$s(t)$	packet	Number of native packets in the t^{th} frame. Its vector form is $\mathbf{s} = [s(1) \quad s(2) \quad \cdots]$.
$a(t)$	N/A	Slope factor for the t^{th} sliding window. Its vector form is $\mathbf{a} = [a(1) \quad a(2) \quad \cdots]$.
H	frame	Horizon length.
$ASP(t)$	N/A	Accumulated sampling probability. It is the total probability accumulated on every packet in frame t through all the sliding windows covering that frame.

Delay-Aware Fountain Codes

DAF codes [10] are delay-aware fountain codes that may improve streaming high quality video over lossy wireless networks. DAF codes deeply integrate channel coding and video coding. In some embodiments, DAF codes include two techniques: time-based sliding window and optimal window-wise sampling strategy.

Time-Based Sliding Window

In some embodiments, a basic idea of DAF codes is to segment the video file into overlapping windows, and then encode and send them consecutively with fountain codes. While the non-overlapping block coding scheme has a relatively small block size, the overlap between sliding windows may allow the decoded packets in a previous window to help the

decoding of future windows. By doing so, the block size may be virtually extended.

In order to maximize block size, the window size W in DAF codes is defined by the maximum tolerable latency T_{Delay} (in the unit of time or number of frames to be transmitted in the window), instead of a fixed number of packets. Therefore this technique is named time-based sliding window. In the rest of this article, we call the window covering the frames of $[t, t+W-1]$ as the t^{th} window.

In [10], the step size between two consecutive windows is Δt. For simplicity, we set $\Delta t = 1$ herein. This is because both the video length T and window size W are defined to be integral multiples of Δt, if $\Delta t > 1$, all the parameters can be down-sampled by a factor of Δt, and all formulas still hold.

Window-wise sampling strategy

Within each window, the fountain codes algorithm may randomly choose the native packets and combine them into a coded packet, according to degree distribution and sampling probability [2]. Because the windows are overlapping, the total sampling probability of a frame is related to all the windows that cover it, as shown in (1).

$$ASP(t) = \sum_{\omega \in \text{allwindowscoverframe}t} p_{\omega}(t), \qquad (1)$$

where $p_{\omega}(t)$ denotes the average sampling probability of each packet in frame t within the window ω. So, $ASP(t)$ denotes the total probability accumulated on every packet in frame t through all the sliding windows covering that frame.

ASP, or *accumulated sampling probability*, is an important metric in DAF codes and will be discussed further herein. The final objective of DAF codes is to minimize the variance of ASP, because the higher efficiency of fountain codes is achieved by a more stable sampling rate.

However, with the time-based sliding window, even if the sampling distribution of every window is uniform, the overall ASP may still be nonuniform. As a result, the sampling probabilities in each window need to be adjusted in order to obtain a uniform overall ASP.

A proper representation of the sampling distribution is required. Storing the entire sampling distribution in each packet header (per-frame description) would be impractical due to both communication and computation overhead. Alternatively, the slope-only description is

proposed: the distribution is approximated by a linear function defined by the slope factor a. The slope factor is a real number ranging from -1 (forming a backward triangular distribution) to 1 (forming a forward triangular distribution). When $a=0$, it represents a uniform distribution.

Given bit rates $\mathbf{s}$ and slope factors $\mathbf{a}$, the resulting ASP for each frame can be computed by (2).

$$ASP_{\mathbf{s},\mathbf{a}}(t)=\mathbf{B}_{\mathbf{s}}(t)\cdot\mathbf{a}+D_{\mathbf{s}}(t), \quad (2)$$

where "$\cdot$" denotes the dot product of the two vectors of $(T-W+1)$ elements, and

$$\begin{aligned}
\mathbf{B}_{\mathbf{s}}(t) &= [B_{\mathbf{s}}(t,1) \quad B_{\mathbf{s}}(t,2)\cdots B_{\mathbf{s}}(t,T-W+1)];\\
B_{\mathbf{s}}(t,i) &= \begin{cases} \dfrac{2\cdot pkt_{\mathbf{s}}(i,t-i+1)-s(t)}{pkt_{\mathbf{s}}(i,W)^2}-\dfrac{1}{pkt_{\mathbf{s}}(i,W)}, \\ \quad if\ i\in[t-W+1,t]; \\ 0, otherwise; \end{cases}\\
D_{\mathbf{s}}(t) &= \sum_{t_0=t-W+1}^{t}\frac{1}{pkt_{\mathbf{s}}(t_0,W)};\\
pkt_{\mathbf{s}}(t,k) &= \sum_{i=t}^{t+k-1}s(i).
\end{aligned} \quad (3)$$

Given the total number of frames T, the window size W, and number of packets in each frame $s(t)$, DAF codes may aim to find a set of slope factors $\mathbf{a}$, for which the variance of the sampling probabilities of all packets attains its minimum value. The optimization problem is defined in (4).

$$\begin{aligned}
&\underset{\mathbf{a}}{\operatorname{argmin}} && \sum_{t=W}^{T-W+1}\left(ASP_{\mathbf{s},\mathbf{a}}(t)-\overline{\mathbf{ASP}_{\mathbf{s},\mathbf{a}}}\right)^2\\
&s.t. && -1\le a_t\le 1, \qquad \forall t,
\end{aligned} \quad (4)$$

where $\overline{\mathbf{ASP}_{\mathbf{s},\mathbf{a}}}=\frac{1}{T-2W+2}\sum_{t=W}^{T-W+1}ASP_{\mathbf{s},\mathbf{a}}(t)$.

Based on the aforementioned two techniques, the inventors have recognized and appreciated two schemes: DAF and the low-complexity version DAF-L.

In-Time Decoding Ratio

As pointed out in [10], a packet decoded after its playback time is useless. This application-specific feature is captured by a new metric called *in-time decoding ratio* (*IDR*),

which only counts the "in-time" decoded packets within the current window. This is in contrast to the *file decoding ratio* (*FDR*) which counts all the decoded packets after a coding session completes.

It has been shown in [10] that DAF and DAF-L schemes yield much higher *IDR* than the existing schemes, such as [7]-[9], [11], [12], and TCP.

Observations of the Inventors

Although DAF and DAF-L can improve the quality of video streaming over lossy wireless network, the inventors have recognized and appreciated areas of improvement:

• The decoding performance of DAF-L may be considered relatively low. Although its performance is higher than other delay-aware schemes, there is still a notable gap between DAF and DAF-L (10% maximum in decoding ratio). That is because DAF-L does not use the window-wise sampling distribution optimization as in DAF. In exchange, DAF-L is a low-complexity and online algorithm.

• Some embodiments may reduce the complexity of DAF in order to improve performance for delay-sensitive video streaming and for long videos. The global optimization function of DAF brings the complexity of $O(T^3)$. Since the computational scale grows cubically with the video length T, the inventors have recognized and appreciated that alternatives may be more practical for long videos.

• DAF may be an offline scheme. Because the bit rate information of all frames is required to perform the optimization, the whole video file may need to be available before being encoded by fountain codes. Therefore, some embodiments of DAF may not be appropriate or possible for live video streaming applications.

Model Predictive Control

The term Model Predictive Control (MPC) does not designate a specific control strategy, but a wide range of methodologies which make an explicit use of a process model to obtain the control signal by minimizing an objective function [15]. The methodology of MPC-based controllers may be characterized by the following steps:

1. At each time τ, the process model is explicitly used to get the output at a future

time horizon H. These predictions $y_\tau(\tau+k)$[1] for $k=1...H$ depend on the known values (history inputs and outputs) up to time τ and the future control signals, which are those to be optimized.

2. The control sequence $u_\tau(\tau+k)$, $k=0...H-1$ is calculated by minimizing the objective function. The minimization function usually takes the form of a quadratic function of the error between the prediction and the target value.

3. The first action $u_\tau(\tau)$ from the calculated control sequence is sent to the process whilst the other control actions are rejected. Then, the next sampling instant $y(\tau+1)$ will be known, and step 1 will be repeated with this new value, and the next control action $u_{\tau+1}(\tau+1)$ is calculated, which in principle will be different to $u_\tau(\tau+1)$ because of the new information available.

The algorithm is iteratively executed each time a new instant is sampled. Various MPC algorithms may only differ with each other in the horizon length, predictive model, and the objective function to be minimized. Note that it is in general not a globally optimal algorithm, since the control decisions are made only based on the history values and the prediction of a finite future horizon.

MPC-Based Optimal Control for DAF Codes

In order to extend the application of DAF codes to live video streaming, the inventors have recognized and appreciated that a method to close the gap between DAF-L and DAF may be valuable: an online optimization algorithm with a lower computational complexity than DAF, but higher performance than DAF-L. Meanwhile, MPC provides a locally optimal online solution to any objective-function-minimizing process control problem, which the inventors have recognized and appreciated as supporting a solution to problems described herein.

MPC-Based DAF Codes: The Offline Version

The inventors have recognized and appreciated that integrating MPC and DAF codes to impose a finite horizon to the optimization process of DAF codes may improve DAF codes at lesast in the areas described above. We call the proposed scheme *DAF-M*, where M stands for

[1] This notation indicates the value of the variable at the time $\tau+k$ is calculated at time τ.

MPC. A general flowchart of some embodiments of DAF-M is shown in FIG. 4. The text on the top of each block describes the procedure in MPC. The text in the parentheses below explains its function in DAF-M.

In this part, we focus on the localization of the optimization problem, so we temporarily assume that *we know the full bit rate information of a video sequence*. As a result, DAF-M may be considered an offline algorithm. The prediction of bit rate and other practical problems for online algorithms will be discussed further below.

Horizon

Denote τ as the index of current time (also the index of current sliding window). Since the window size is W, the frame that was newly added to the encoding buffer is the $(\tau+W-1)^{\text{th}}$ one.

In order to determine what slope to use in the τ^{th} window, we may calculate the best set of slopes over a future horizon based on both the known ASP in the past and the predicted ASP in the future, then take the first slope as the one used in encoding the τ^{th} window. We denote the length of the horizon as H, so the range of windows involved in our slope optimization is $[\tau, \tau+H-1]$.

We hope to clarify the difference between window size W and horizon length H here. The two concepts are confusable, since both of them are spans of frames, and with both values increasing, the performance and computational complexity of the controller tend to be higher. In fact, they are independent parameters that serve different purposes. W is typically chosen in accordance with the longest tolerable end-to-end delay. It is specified by the user, so it is an input value that does not change in a communication session. On the other hand, H is a parameter to balance the computational complexity and desired performance. It is chosen by the application according to the computing power, network condition, quality demand, etc. The extreme case would be H equals to 1, with which only one slope is calculated, the algorithm reduces to a greedy algorithm; if H equals to $T-\tau+1$, thus all the frames in the future are considered in order to obtain each slope, it becomes the same as the globally optimal algorithm of DAF.

Objective Function

The inventors have recognized and appreciated that a good target may be similar to the global optimization problem in (4), which is minimizing the variance of ASP. With a finite

horizon imposed, the objective function is limited to a local range. Because the slopes in the horizon $[\tau, \tau+H-1]$ can affect the ASP of the frames in $[\tau, \tau+H+W-2]$, the objective function is the variance over that range as in (5).

$$J = \sum_{t=\tau}^{\tau+H+W-2} \left(P_{\mathbf{s},\mathbf{a}_\tau^H}(t) - \overline{\mathbf{P}_{\mathbf{s},\mathbf{a}_\tau^H}} \right)^2, \qquad (5)$$

where $\mathbf{a}_\tau^H$ denotes the H-length slope vector for the windows starting from τ. $\mathbf{P}$ denotes the vector of ASP in the range of $[\tau, \tau+H+W-2]$, including both past and predictive values. The calculation of $\mathbf{P}$ will be discussed in the following part. $\overline{\mathbf{P}_{\mathbf{s},\mathbf{a}_\tau^H}}$ is the average value over $\mathbf{P}$.

Process Model

In some embodiments, the process model may play a decisive role in a MPC controller. The chosen model may need to be capable of capturing the process dynamics so as to precisely predict the future outputs. Different from the problems in most industrial control scenarios, in DAF codes, if the bit rate vector $\mathbf{s}$ is given, the mapping between control sequences (slope vector) and the predicted output (future ASP) may be deterministic:

$$\mathbf{ASP}(\mathbf{s},\mathbf{a}) = [ASP_{\mathbf{s},\mathbf{a}}(1) \quad ASP_{\mathbf{s},\mathbf{a}}(2) \cdots ASP_{\mathbf{s},\mathbf{a}}(l)], \qquad (6)$$

where $\mathbf{s}$ is a bit rate vector, and $\mathbf{a}$ is a slope vector. $ASP_{\mathbf{s},\mathbf{a}}(t)$ is defined in (2). Vector length l is equal to the length of $\mathbf{a}$. In the calculation of ASP, $\mathbf{s}$ needs to contain $W-1$ more elements than $\mathbf{a}$.

Although ASP model is deterministic if we know all the bit rates $\mathbf{s}$ and slope parameters $\mathbf{a}$ in the future, the process model may no longer be deterministic when a finite horizon is imposed.

For the current window τ, the ASP in the range of objective function in (5) consists of three parts, all of which may be $(H+W-1)$-length vectors representing a part of ASP within the range of $[\tau, \tau+H+W-2]$. Some of the components only have a part of non-zero elements in that range, which will be pointed out later.

- $\mathbf{P}_{init}$: the initial ASP that already sampled by previous windows.

  Non-zero range: $[\tau, \tau + W - 2]$.

$$\mathbf{P}_{init} = \mathbf{ASP}\left(\mathbf{s}_{\tau-W+1}^{2W-2}, \mathbf{a}_{\tau-W+1}^{W-1}\right), \qquad (7)$$

where the index notation in $\mathbf{s}_t^l$ and $\mathbf{a}_t^l$ means the elements in these two vectors are the t^{th} to $(t+l-1)^{\text{th}}$ elements from vectors $\mathbf{s}$ and $\mathbf{a}$, respectively. Because $\mathbf{P}_{init}$ is computed based on known $\mathbf{s}$ and previous selections of $\mathbf{a}$, it may have no variable parameter.

- $\tilde{\mathbf{P}}$: the range where ASP are affected by the currently optimized slopes within the horizon.

  Non-zero range: $[\tau, \tau + H + W - 2]$.

$$\tilde{\mathbf{P}}_{\mathbf{s},\mathbf{a}_\tau^H} = \mathbf{ASP}\left(\mathbf{s}_\tau^{H+W-1}, \mathbf{a}_\tau^H\right), \qquad (8)$$

where $\mathbf{a}_\tau^H$ is the H -length slope vector to be optimized as in (5).

- $\hat{\mathbf{P}}$: the range where ASP will be affected by the slopes out of the horizon range in the future.

  Non-zero range: $[\tau + H, \tau + H + W - 2]$.

$$\hat{\mathbf{P}}_{\mathbf{s}} = \mathbf{ASP}\left(\mathbf{s}_{\tau+H}^{2W-2}, \hat{\mathbf{a}}_{\tau+H}^{W-1}\right), \qquad (9)$$

where $\hat{\mathbf{a}}$ denotes the prediction of future slopes. Since the optimal choices of the slopes in this part will also be affected by the slopes in the further future, in order to limit the length of horizon, we shall make a prediction without full knowledge about further frames. This is the part that may bring error. Fortunately, because the selection of slopes of frames from further away will have less impact on current optimal slope choice, a simple and safe prediction may be enough. In some embodiments, an all-zero vector may be used as $\hat{\mathbf{a}} = \mathbf{0}$, assuming all the windows in that range will use uniform distributions. In that case, we can eliminate the variable parameter $\hat{\mathbf{a}}$. The future bit rates up to time $\tau + H + 2W - 3$ may still be needed as input parameters.

The result is a combination of the three components:

$$\mathbf{P}_{\mathbf{s},\mathbf{a}_\tau^H} = \mathbf{P}_{init} + \tilde{\mathbf{P}}_{\mathbf{s},\mathbf{a}_\tau^H} + \hat{\mathbf{P}}_{\mathbf{s}}. \qquad (10)$$

We plug (6) - (10) into the objective function (5) and solve the optimization function (11).

$$\begin{aligned} \underset{\mathbf{a}_\tau^H}{\text{argmin}} \quad & J = \sum_{t=\tau}^{\tau+H+W-2} \left(P_{\mathbf{s},\mathbf{a}_\tau^H}(t) - \overline{\mathbf{P}_{\mathbf{s},\mathbf{a}_\tau^H}} \right)^2, \\ s.t. \quad & -1 \le a_t \le 1, \qquad \forall t. \end{aligned} \qquad (11)$$

After solving the optimal $\mathbf{a}_\tau^H$, only the first element $a_\tau(\tau)$ may be chosen to encode the τ^{th} window.

Complexity

For each window τ, the optimal slope (11) can be solved by Karush-Kuhn-Tucker (KKT) conditions. Because there are H variables to optimize and $2H$ conditions for KKT conditions, the optimization process yields a system of equations with $3H$ equations. If we omit constant factors, the solution involves the generation of a parameter matrix of $H \times H$ and the computation of its inverse matrix. The algorithm will be executed iteratively for each window for T times. As a result, the total computational complexity is $O(T \times H^3)$. Since $H = T$ and H does not increase with video length, it can be considered as a linear complexity algorithm for time T. Comparatively, DAF-M may have a much lower complexity than $O(T^3)$ of DAF.

Online Algorithm With Bit Rate Prediction

From (8) and (9), we know that in order to optimize the slopes in horizon H, we may need to know the bit rates in $H + 2W - 3$ frames ahead of current time τ, or $H + W - 2$ frames ahead of the last known (a.k.a. the $(\tau + W - 1)^{\text{th}}$) frame. However, in live-stream videos, we do not know the future bit rates beforehand. As a result, in order to make DAF-M an online algorithm, the ivnentors have recognized and appreciated that a prediction may need to be made on future bit rates, and the length of prediction may be $H + W - 2$.

Admittedly, a more accurate prediction of video bit rate will help MPC optimization to obtain better control sequences. Therefore, using advanced algorithms, such as [18]-[22], may

improve the coding result. A survey of video frame size forecasting algorithms can be found in [23]. Nevertheless, since video bit rate is a stochastic process, we can never expect a perfect prediction. Since our work does not focus on bit rate prediction algorithms, a simple linear formula (12) is adapted to predict the future bit rates. A similar formula is used in [24] and yields good results.

$$\hat{s}_{t+1} = (1-\alpha)\hat{s}_t + \alpha s_t, \quad (12)$$

where s_t denotes the bit rate of the frame newly added to the DAF encoder, and $\hat{s}_{t+1}$ denotes the prediction for the future frames. $\alpha \in (0,1]$ controls the weight of new frame in the prediction of bit rate. We fix it to 0.75 so our algorithm can get good average performances in our implementations.

In some embodiments, the DAF encoder may be implemented in an ASIC, an FPGA, or a similar integrated circuit chip. Alternatively or additionally, the system may be implemented as a driver controlling network interface hardware or in any other suitable way.

Since (12) is not accurate enough to perform the frame-by-frame prediction, only a level of future bit rate is obtained. We replace all the elements of $\mathbf{s}$ in (10) which exceed the last known frame with the predicted bit rate $\hat{s}_{t+1}$. We call the online MPC-based DAF codes algorithm as *DAF-O*.

When it comes to computational complexity, in some embodiments, the prediction of bit rate in DAF-O is only $O(1)$, so the online algorithm may not increase the total computational complexity in DAF-M.

Optimization Results and Evaluations

In this part, we show some examples of optimization results obtained by different schemes, and analyze the resulting ASP.

ASP Results Comparison

FIG. 5a shows the bit rates for CIF video sequence *foreman* coded with H.264/AVC standard. FIG. 5b-d present the optimization results using DAF-O, DAF-M, and DAF, respectively, with $W = 30$ and $H = 10$. In FIG. 5b-d, each slope line represents a sampling distribution within a window. Since $\Delta t = 1$, there should be a window for every $[t, t+W-1]$ span. Because there are too many windows to be clearly shown in one figure, only a fraction of

the windows is presented here.

The inventors have recognized and appreciated that (i) the first $W-1$ slopes in DAF-M (FIG. 5c) and DAF-O (FIG. 5b) are uniform distributions. That is because there may not be enough past ASP to calculate $\mathbf{P}_{init}$ in that range (see (7)), so the slopes in that range are set to 0 as a warm-up period; (ii) the latter part of DAF-M is very similar to that in the globally optimal result of DAF in FIG. 5d, whilst DAF-O is not, because of its lack of information about the future bit rates.

FIG. 6 shows the resulting ASP using the optimized slopes from FIG. 5b-d (including DAF-L, which uses all uniform distributions). The numbers following scheme names in the legend are corresponding variations of ASP, and there is

$$\mathrm{DAF} < \mathrm{DAF-M} < \mathrm{DAF-O} < \mathrm{DAF-L}.$$

Measuring the Quality of ASP

Although we use variance of ASP as the objective function in (5), the inventors have recognized and appreciated that it may not be an effective metric to measure the quality of the resulting ASP in fountain codes. For example, if obtained sampling distributions generate very low ASP, even if the variance is low, they are not desirable. Besides, the performance of obtained sampling distribution may vary under different code rates. Therefore, we introduce a new metric to measure the quality of ASP, called *ASP coverage ratio*, denoted as ρ as in (13).

$$\rho_{\mathbf{P}}(c) = \frac{\sum_{t=1}^{T} \left[P_t \geq \frac{1}{c} \right]}{T}, \quad (13)$$

where $[\cdot]$ is an Iverson bracket notation, which is defined to be 1 if the condition in square brackets is satisfied, and 0 otherwise. $\rho(c)$ gives the ratio of frames whose ASP surpasses $1c$. It is a monotonically increasing function, and its output values are in the range of $[0,1]$: if $c < 1P_{max}$, the output value is 0; if $c \geq 1P_{min}$, the output value is 1 (where P_{max} and P_{min} denote the maximum and minimum values in $\mathbf{P}$ respectively).

The inventors have recognized and appreciated that it is a good metric to measure the quality of ASP in this case for the following reasons. ASP reflects the average accumulated sampling probability for a frame, so, under the same code rate, greater ASP of a frame indicates higher probability for it to be sampled than others. It also means, in order to be sampled into at

least one coded packet, a frame with higher ASP expects lower code rate, and ASP is inversely proportional to expected code rate. Therefore, if the ASP of a frame is higher than the reciprocal of code rate $1c$, that frame is likely to be sampled in coded packets under code rate c, thus it has the chance to be decoded on the receiver side. As a result, $\rho(c)$ represents the ratio of frames that could be decoded under code rate c.

FIG. 7 shows the coverage ratio curves of ASP from FIG. 6 and block coding. c is normalized to the range of $[0,1]$. The inventors have recognized and appreciated that the result justifies the gains brought by the proposed DAF codes: when c is greater than 60% of $1P_{min}$, the ASP coverage ratios are over 60% and there is $\mathrm{DAF} \geq \mathrm{DAF-M} \geq \mathrm{DAF-O} \geq \mathrm{DAF-L}$. That means in terms of the ratio of possible decoded frames within that code rate range, in some embodiments, DAF outperforms DAF-M, DAF-M outperforms DAF-O, and DAF-O outperforms DAF-L.

Conversely, the relationships of ρ are inversed when $c < 50\%$. However, since in those scenarios the code rates are too low that the decoding ratios are below 50% and the video can hardly be viewed for all schemes, they are not the cases of most concern to users. In other words, the proposed DAF codes sacrifice the performance when the code rate is extremely low, for better performance when the code rate is in a moderately high range.

It should be noted that this metric does not fully represent the actual decoding ratio in video communication. It only considers the sampling probability, but (i) it does not imply when the frames are decoded, so it does not represent in-time decoding ratio (*IDR*); (ii) it does not consider the factors of fountain codes, such as degree distribution, codeword length, etc. That is the reason why the ρ curve of block coding does not look significantly worse than DAF-L in FIG. 7, while the *IDR* results of block coding is far inferior to any other schemes, as we will see in the next section.

Exemplary Experimental Results

Simulator Setup

The encoding/decoding parts of some embodiments may generally follow the DAF framework in [10], with changes made to the Video Preprocessing Module, especially with implementation of a different optimization algorithm in some embodiments.

We conduct the simulation experiments on Common Open Research Emulator (CORE)

[25] and Extendable Mobile Ad-hoc Network Emulator (EMANE) [26]. The working environment is set up on Oracle® VM VirtualBox virtual machines.

We use CORE to emulate the topology of the virtual network and the relay nodes. Two VMs are connected to the virtual network as a source (or encoder/sender) node running the client application, and a destination (or decoder/receiver) node running the server application. A video is streamed from client to server using different schemes.

EMANE is used for emulation of IEEE 802.11b on PHY and MAC layer of each wireless node. Because of the forward error correction (FEC) nature of fountain codes, we disable the retransmission mechanism of 802.11b for all fountain-code-based schemes. The adaptive rate selection mechanism of 802.11b is also disabled. Ad-hoc On-Demand Distance Vector (AODV) protocol is used for routing.

There are two nodes in the network: a source node and a destination node. The communication path from the source to the destination has one hop. The distance between the two nodes is carefully set so that the packets with 1024-byte payload will experience a 10% loss rate.

Performance Evaluation

We implement eight schemes for comparisons, which are abbreviated as follows:

1. *DAF*: the full optimization version of delay-aware fountain codes in [10].
2. *DAF-M*: the MPC-based DAF codes as described herein.
3. *DAF-O*: the online version of MPC-based DAF codes as described herein.
4. *DAF-L*: the non-optimized version of DAF codes. All the windows may use uniform distributions in some embodiments.
5. *S-LT*: the sliding window LT code from [8].
6. *Block*: the block coding for fountain codes.
7. *Expand*: the expanding window scheme of [11].
8. *TCP*: uses TCP protocol to stream video. In order to add delay awareness, the video file is also segmented into the blocks like in "Block" scheme, but they are sent using TCP.

All the seven fountain-code-based schemes use the following parameter setting: the packet size $P = 1024$ bytes; for degree distribution, let $\delta = 0.02$, $c = 0.4$ (as defined in [2, Definition 11]). For TCP scheme, the maximum data rate is limited to the same amount as

required by the fountain-code-based schemes for the sake of fairness.

Several benchmark CIF test sequences, provided by [27], are used for our evaluation. They are coded into H.264/AVC format using *x264* [28], encapsulated into ISO MP4 files using *MP4Box* [29], and streamified by *mp4trace* tool from *EvalVid* tool-set [30]. The coding structure is IPPP, which contains only one I-frame (the first frame) and no B-frame, and all the rest are P-frames. All the delays shown in the experiments are in the unit of seconds. Denote by C and T_{Delay} the code rate and the tolerable delay, respectively.

We conduct 20 experiments for each setting with different random seeds, and take the median value of them as the performance measure. The in-time decoding ratio (IDR) results are presented since it is a better metric in live video streaming than file decoding ratio.

Evaluate The Effect Of Horizon Length

In this case, we compare the performance of four DAF-based schemes. For DAF-M and DAF-O, we set H to 1, 5, 10, 15 and 20 to observe the influence of horizon length.

FIG. 8 shows the IDR curves of the DAF-based schemes for *foreman*. Because there are two dimensions of variables, we obtain the IDR curves vs. C when fixing $T_{Delay} = 1.2\text{s}$ in FIG. 8a, and also obtain the IDR curves vs. T_{Delay} when fixing $C = 0.77$ in FIG. 8b. Results of DAF-M and DAF-O are plotted with dashed lines and dotted lines, resepectively. The different horizon lengths H are indicated by the width of lines and the brightness of colors: the thicker and darker lines indicate larger H , and the thinner and lighter lines indicate smaller H . Various values of H may be chosen for MPC-O and MPC-M.

The numerical results of them when fixing both delay $T_{Delay} = 1.2\text{s}$ and code rate $C = 0.8$ for *foreman* are shown in Table 2.

Table 2: *IDR* comparisons of DAF-based schemes using different horizon lengths.

Scheme	*H*	*IDR*
DAF	N/A	93.71%
DAF-M	20	90.16%
	15	88.45%
	10	87.80%
	5	86.15%
	1	80.86%
DAF-O	20	85.29%
	15	83.81%
	10	87.03%
	5	82.88%
	1	68.00%
DAF-L	N/A	76.52%

From the results above, the inventors have recognized and appreciated the following:

• The decoding ratio of all the schemes is an increasing function of T_{Delay}, and also a decreasing function of C. That means larger delay and lower code rate lead to higher overall performance, which meets our expectation.

• Among the DAF-based schemes, the performance of DAF may be the highest in some embodiments, while DAF-L may be the lowest in some embodiments. The results of DAF-M and DAF-O may generally be between those of DAF and DAF-L in some embodiments. The relationships of decoding ratios is consistent with the ASP coverage ratios as seen in FIG. 7.

• In general, DAF-M may outperform DAF-O under the same horizon length. That is because DAF-M can be deemed as DAF-O with "perfect" bit rate prediction. As a result, the gap between DAF and DAF-M is only caused by local bit rate knowledge, while the performance drop from DAF to DAF-O may come from both limited horizon length and inaccuracy of bit rate prediction.

• For DAF-M, the decoding ratio gets higher with the increasing horizon length. However, since the computational complexity increases cubically with H, an extra large H may not be affordable.

• For DAF-O, longer horizon length *may not guarantee* better performance. In the example of Table 2, the highest IDR of DAF-O is achieved when $H = 10$, but not $H = 20$. The reason is that in DAF-O, the long-term bit rate prediction may become inaccurate. Due to accumulated long-term prediction error, the performance of DAF-O may reduce when the prediction length increases.

• When $H = 1$, the performance of DAF-O is significantly lower than other H values. That is because it is deducted to a greedy algorithm, and the decision is only based on the history ASP.

Compare With the Existing Schemes

In this case, we compare the proposed schemes with the existing video streaming algorithms. To strike a good balance between complexity and performance, we use $H = 10$ for MPC-O and MPC-M.

First, we want to show IDR comparisons of the *online* window-based fountain codes schemes: DAF-O, DAF-L, S-LT, expanding window and block coding. We choose all the combinations of $T_{Delay} \in [0.6,1.5]$ and $C \in [0.6,0.9]$ to conduct the experiments on *foreman*. There are two dimensions of variables, so the results of each scheme form a surface of IDR. FIG. 9 shows five surfaces of the online schemes.

The numerical results obtained by more schemes for more sequences are shown in Table 3.

Table 3: *IDR* comparisons of proposed schemes and existing video streaming algorithms.

Sequence	Code Rate	Delay (s)	Scheme	IDR
foreman	0.72	1	DAF	96.72%
			DAF-M	96.39%
			DAF-O	95.87%
			DAF-L	92.62%
			S-LT	76.65%
			Expand	59.90%
			Block	35.05%
			TCP	72.74%
coastguard	0.72	0.7	DAF	95.78%
			DAF-M	95.05%
			DAF-O	94.76%
			DAF-L	94.13%
			S-LT	70.97%
			Expand	47.72%
			Block	32.58%
			TCP	65.45%
mobile	0.86	1.3	DAF	91.30%
			DAF-M	91.08%
			DAF-O	90.48%
			DAF-L	89.14%
			S-LT	79.79%
			Expand	50.42%
			Block	13.21%
			TCP	76.43%

From the results above, we have the following observations:

• Among all the schemes, DAF may have the highest performance, followed by DAF-M, both of which are offline algorithms. Considering the computational complexity of DAF-M may be orders of magnitude lower than DAF in some embodiments, DAF-M may be a more practical offline algorithm in some embodiments.

• Among all the online schemes, DAF-O may have the highest performance, followed by DAF-L. As shown in FIG. 9, the surface of DAF-O is almost always higher than any other scheme.

• If C is too high or T_{Delay} is too small, the performance of DAF-L may be lower than DAF. The result accords with the ASP coverage ratio in FIG. 8 when c is small, but it is not a very typical scenario since all of them are too low to be properly watched.

• The performance of TCP is relatively low. The reason is that TCP is not suitable for

wireless channels where packet loss rate is high [31]. Its congestion control mechanism does not help the performance.

• Block coding scheme performs the lowest among all schemes, although its ASP coverage ratio is not very low compared to others. The reason is that the blocks are too small and non-overlapping, so the coding overhead is very large [13].

References

The following references are incorporated herein by reference in their entireties:

[1] J. W. Byers, M. Luby, and M. Mitzenmacher, "A digital fountain approach to asynchronous reliable multicast," IEEE Journal on Selected Areas in Communications, vol. 20, no. 8, pp. 1528–1540, 2002.

[2] M. Luby, "LT codes," in 2013 IEEE 54th Annual Symposium on Foundations of Computer Science. IEEE Computer Society, 2002, pp. 271–271.

[3] A. Shokrollahi, "Raptor codes," IEEE Transactions on Information Theory, vol. 52, no. 6, pp. 2551–2567, 2006.

[4] M. Luby, A. Shokrollahi, M. Watson, T. Stockhammer, and L. Minder, "RaptorQ forward error correction scheme for object delivery (RFC 6330)," IETF Request For Comments, 2011.

[5] Q. Huang, K. Sun, X. Li, and D. O. Wu, "Just FUN: A joint fountain coding and network coding approach to loss-tolerant information spreading," in Proceedings of the 15th ACM international symposium on Mobile ad hoc networking and computing. ACM, 2014, pp. 83–92.

[6] J.-P. Wagner, J. Chakareski, and P. Frossard, "Streaming of scalable video from multiple servers using rateless codes," in IEEE International Conference on Multimedia and Expo, 2006. IEEE, 2006, pp. 1501–1504.

[7] S. Ahmad, R. Hamzaoui, and M. M. Al-Akaidi, "Unequal error protection using fountain codes with applications to video communication," IEEE Transactions on Multimedia, vol. 13, no. 1, pp. 92–101, 2011.

[8] M. C. Bogino, P. Cataldi, M. Grangetto, E. Magli, and G. Olmo, "Sliding-window digital fountain codes for streaming of multimedia contents," in IEEE International Symposium on Circuits and Systems, 2007. ISCAS 2007. IEEE, 2007, pp. 3467–3470.

[9] P. Cataldi, M. Grangetto, T. Tillo, E. Magli, and G. Olmo, "Sliding-window raptor codes for efficient scalable wireless video broadcasting with unequal loss protection," IEEE

Transactions on Image Processing, vol. 19, no. 6, pp. 1491–1503, 2010.

[10] K. Sun, H. Zhang, and D. Wu, "DAF codes: Delay-aware fountain codes for video streaming," IEEE Transactions on Multimedia, vol. 20, no. 8, pp. 1528–1540, 2016.

[11] D. Sejdinovic, D. Vukobratovic, A. Doufexi, V. Senk, and R. J. Piechocki, "Expanding window fountain codes for unequal error protection," IEEE Transactions on Communications, vol. 57, no. 9, pp. 2510–2516, 2009.

[12] D. Vukobratovic, V. Stankovic, D. Sejdinovic, L. Stankovic, and Z. Xiong, "Scalable video multicast using expanding window fountain codes," IEEE Transactions on Multimedia, vol. 11, no. 6, pp. 1094–1104, 2009.

[13] G. Liva, E. Paolini, and M. Chiani, "Performance versus overhead for fountain codes over Fq," IEEE Communications Letters, vol. 14, no. 2, pp. 178–180, 2010.

[14] K. Sun and D. Wu, "Video rate control strategies for cloud gaming," Journal of Visual Communication and Image Representation, vol. 30, pp. 234–241, 2015.

[15] E. F. Camacho and C. B. Alba, Model predictive control. Springer Science & Business Media, 2013.

[16] H. Borhan, A. Vahidi, A. M. Phillips, M. L. Kuang, I. V. Kolmanovsky, and S. D. Cairano, "MPC-based energy management of a powersplit hybrid electric vehicle," IEEE Transactions on Control Systems Technology, vol. 20, no. 3, pp. 593–603, 2012.

[17] P. Falcone, H. Eric Tseng, F. Borrelli, J. Asgari, and D. Hrovat, "MPCbased yaw and lateral stabilisation via active front steering and braking," Vehicle System Dynamics, vol. 46, no. S1, pp. 611–628, 2008.

[18] A. Abdennour, "Short-term MPEG-4 video traffic prediction using ANFIS," International Journal of Network Management, vol. 15, no. 6, pp. 377–392, 2005.

[19] A. D. Doulamis, N. D. Doulamis, and S. D. Kollias, "An adaptable neural-network model for recursive nonlinear traffic prediction and modeling of MPEG video sources," IEEE Transactions on Neural Networks, vol. 14, no. 1, pp. 150–166, 2003.

[20] A. Bhattacharya, A. G. Parlos, and A. F. Atiya, "Prediction of MPEG-coded video source traffic using recurrent neural networks," IEEE Transactions on Signal Processing, vol. 51, no. 8, pp. 2177–2190, 2003.

[21] K. Y. Lee, M. Kim, H.-S. Jang, and K.-S. Cho, "Accurate prediction of real-time MPEG-4 variable bit rate video traffic," ETRI journal, vol. 29, no. 6, pp. 823–825, 2007.

[22] K. Sun and B. Yan, "Efficient P-frame complexity estimation for frame layer rate control of H.264/AVC," in 18th IEEE International Conference on Image Processing (ICIP),

2011. IEEE, 2011, pp. 1669–1672.

[23] R. J. Haddad, M. P. McGarry, and P. Seeling, "Video bandwidth forecasting," IEEE Communications Surveys & Tutorials, vol. 15, no. 4, pp. 1803–1818, 2013.

[24] A. M. Adas, "Using adaptive linear prediction to support real-time VBR video under RCBR network service model," IEEE/ACM Transactions on Networking, vol. 6, no. 5, pp. 635–644, 1998.

[25] J. Ahrenholz, "Comparison of core network emulation platforms," in MILITARY COMMUNICATIONS CONFERENCE, 2010-MILCOM 2010. IEEE, 2010, pp. 166–171.

[26] U.S. Naval Research Laboratory, Networks and Communication Systems Branch. The extendable mobile ad-hoc network emulator (EMANE). [Online]. Available: http://www.nrl.navy.mil/itd/ncs/products/emane

[27] P. Seeling and M. Reisslein, "Video transport evaluation with H.264 video traces," IEEE Communications Surveys and Tutorials, vol. 14, no. 4, pp. 1142–1165, 2012.

[28] x264 team. x264. [Online]. Available: http://www.videolan.org/developers/x264.html

[29] GPAC project. MP4Box. [Online]. Available: http://gpac.wp.minestelecom.fr/mp4box/

[30] J. Klaue, B. Rathke, and A. Wolisz, "Evalvid–a framework for video transmission and quality evaluation," in Computer Performance Evaluation. Modelling Techniques and Tools. Springer, 2003, pp. 255–272.

[31] G. Holland and N. Vaidya, "Analysis of TCP performance over mobile ad hoc networks," Wireless Networks, vol. 8, no. 2/3, pp. 275–288, 2002.

Computing Environment

Techniques for increasing data throughput and decreasing transmission delay along a data link from a source node to a sink node may be implemented on any suitable hardware, including a programmed computing system. For example, FIG. 1 illustrates a system implemented with multiple computing devices, which may be distributed and/or centralized. Also, FIGS. 2 and 3 illustrate algorithms executing on at least one computing device. FIG. 19 illustrates an example of a suitable computing system environment 300 on which embodiments of these algorithms may be implemented. This computing system may be representative of a computing system that implements the described technique of increasing data throughput and decreasing transmission delay from a source node to a sink node via a relay node. However, it

should be appreciated that the computing system environment 300 is only one example of a suitable computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the invention. Neither should the computing environment 300 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary operating environment 300.

The invention is operational with numerous other general purpose or special purpose computing system environments or configurations. Examples of well-known computing systems, environments, and/or configurations that may be suitable for use with the invention include, but are not limited to, personal computers, server computers, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, set top boxes, programmable consumer electronics, network PCs, minicomputers, mainframe computers, distributed computing ,environments, or cloud-based computing environments that include any of the above systems or devices, and the like.

The techniques described herein may be implemented in whole or in part within network interface 370. The computing environment may execute computer-executable instructions, such as program modules. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote computer storage media including memory storage devices.

With reference to FIG. 19, an exemplary system for implementing the invention includes a general purpose computing device in the form of a computer 310. Though a programmed general purpose computer is illustrated, it should be understood by one of skill in the art that algorithms may be implemented in any suitable computing device. Accordingly, techniques as described herein may be implemented in a system for increasing data throughput and decreasing transmission delay along a data link from a source node to a sink node. These techniques may be implemented in such network devices as originally manufactured or as a retrofit, such as by changing program memory devices holding programming for such network devices or software download. Thus, some or all of the components illustrated in FIG. 19, though illustrated as part of a general purpose computer, may be regarded as representing portions of a node or other component in a network system.

Components of computer 310 may include, but are not limited to, a processing unit 320, a system memory 330, and a system bus 321 that couples various system components including the system memory 330 to the processing unit 320. The system bus 321 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. By way of example and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus also known as Mezzanine bus.

Computer 310 typically includes a variety of computer readable media. Computer readable media can be any available media that can be accessed by computer 310 and includes both volatile and nonvolatile media, removable and non-removable media. By way of example, and not limitation, computer readable media may comprise computer storage media and communication media. Computer storage media includes both volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium that can be used to store the desired information and that can be accessed by computer 310. Communication media typically embodies computer readable instructions, data structures, program modules, or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, radio frequency (RF), infrared (IR), and other wireless media. Combinations of any of the above should also be included within the scope of computer readable media.

The system memory 330 includes computer storage media in the form of volatile and/or nonvolatile memory such as read only memory (ROM) 331 and random access memory (RAM) 332. A basic input/output system 333 (BIOS), containing the basic routines that help to transfer information between elements within computer 310, such as during start-up, is

typically stored in ROM 331. RAM 332 typically contains data and/or program modules that are immediately accessible to and/or presently being operated on by processing unit 320. By way of example and not limitation, FIG. 19 illustrates operating system 334, application programs 335, other program modules 336, and program data 337.

The computer 310 may also include other removable/non-removable, volatile/nonvolatile computer storage media. By way of example only, FIG. 19 illustrates a hard disk drive 341 that reads from or writes to non-removable, nonvolatile magnetic media, a magnetic disk drive 351 that reads from or writes to a removable, nonvolatile magnetic disk 352, and an optical disk drive 355 that reads from or writes to a removable, nonvolatile optical disk 356 such as a CD-ROM or other optical media. Other removable/non-removable, volatile/nonvolatile computer storage media that can be used in the exemplary operating environment include, but are not limited to, magnetic tape cassettes, flash memory cards, digital versatile disks, digital video tape, solid state RAM, solid state ROM, and the like. The hard disk drive 341 is typically connected to the system bus 321 through an non-removable memory interface such as interface 340, and magnetic disk drive 351 and optical disk drive 355 are typically connected to the system bus 321 by a removable memory interface, such as interface 350.

The drives and their associated computer storage media discussed above and illustrated in FIG. 19, provide storage of computer readable instructions, data structures, program modules, and other data for the computer 310. In FIG. 19, for example, hard disk drive 341 is illustrated as storing operating system 344, application programs 345, other program modules 346, and program data 347. Note that these components can either be the same as or different from operating system 334, application programs 335, other program modules 336, and program data 337. Operating system 344, application programs 345, other program modules 346, and program data 347 are given different numbers here to illustrate that, at a minimum, they are different copies. A user may enter commands and information into the computer 310 through input devices such as a keyboard 362 and pointing device 361, commonly referred to as a mouse, trackball, or touch pad. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 320 through a user input interface 360 that is coupled to the system bus, but may be connected by other interface and bus structures, such as a parallel port, game port, or a universal serial bus (USB). A monitor 391 or other type of display device is also connected to the system bus 321 via an interface, such as a video

interface 390. In addition to the monitor, computers may also include other peripheral output devices such as speakers 397 and printer 396, which may be connected through an output peripheral interface 395.

The computer 310 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 380. The remote computer 380 may be a personal computer, a server, a router, a network PC, a peer device, or some other common network node, and typically includes many or all of the elements described above relative to the computer 310, although only a memory storage device 381 has been illustrated in FIG. 19. The logical connections depicted in FIG. 19 include a local area network (LAN) 371 and a wide area network (WAN) 373, but may also include other networks. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets, and the Internet.

When used in a LAN networking environment, the computer 310 is connected to the LAN 371 through a network interface or adapter 370. When used in a WAN networking environment, the computer 310 typically includes a modem 372 or other means for establishing communications over the WAN 373, such as the Internet. The modem 372, which may be internal or external, may be connected to the system bus 321 via the user input interface 360, or other appropriate mechanism. In a networked environment, program modules depicted relative to the computer 310, or portions thereof, may be stored in the remote memory storage device. By way of example and not limitation, FIG. 19 illustrates remote application programs 385 as residing on memory device 381. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

Having thus described several aspects of at least one embodiment of this invention, it is to be appreciated that various alterations, modifications, and improvements will readily occur to those skilled in the art.

In some embodiments, techniques described herein may be used in streaming multimedia services. Such services may include streaming multimedia, such as video and/or audio, over a network such as the Internet between a streaming server and a client. Some embodiments of a streaming service may be over a different network, such as a LAN, where the streaming server may be installed on a computer within the premises of a customer, such as a house or office building. Alternatively or additionally, the streaming server may be geographically remote relative to the clients, and the connection between the server and clients

may be a dedicated wireless connection. Alternatively, the connection may be over a shared network such as a 5G cellular network.

Such alterations, modifications, and improvements are intended to be part of this disclosure, and are intended to be within the spirit and scope of the invention. Further, though advantages of the present invention are indicated, it should be appreciated that not every embodiment of the invention will include every described advantage. Some embodiments may not implement any features described as advantageous herein and in some instances. Accordingly, the foregoing description and drawings are by way of example only.

The above-described embodiments of the present invention can be implemented in any of numerous ways. For example, the embodiments may be implemented using hardware, software or a combination thereof. When implemented in software, the software code can be executed on any suitable processor or collection of processors, whether provided in a single computer or distributed among multiple computers. Such processors may be implemented as integrated circuits, with one or more processors in an integrated circuit component. Though, a processor may be implemented using circuitry in any suitable format.

Further, it should be appreciated that a computer may be embodied in any of a number of forms, such as a rack-mounted computer, a desktop computer, a laptop computer, or a tablet computer. Additionally, a computer may be embedded in a device not generally regarded as a computer but with suitable processing capabilities, including a Personal Digital Assistant (PDA), a smart phone, or any other suitable portable or fixed electronic device.

Also, a computer may have one or more input and output devices. These devices can be used, among other things, to present a user interface. Examples of output devices that can be used to provide a user interface include printers or display screens for visual presentation of output and speakers or other sound generating devices for audible presentation of output. Examples of input devices that can be used for a user interface include keyboards, and pointing devices, such as mice, touch pads, and digitizing tablets. As another example, a computer may receive input information through speech recognition or in other audible format.

Such computers may be interconnected by one or more networks in any suitable form, including as a local area network or a wide area network, such as an enterprise network or the Internet. Such networks may be based on any suitable technology and may operate according to any suitable protocol and may include wireless networks, wired networks, or fiber optic networks.

Also, the various methods or processes outlined herein may be coded as software that is executable on one or more processors that employ any one of a variety of operating systems or platforms. Additionally, such software may be written using any of a number of suitable programming languages and/or programming or scripting tools, and also may be compiled as executable machine language code or intermediate code that is executed on a framework or virtual machine.

In this respect, the invention may be embodied as a computer readable storage medium (or multiple computer readable media) (e.g., a computer memory, one or more floppy discs, compact discs (CD), optical discs, digital video disks (DVD), magnetic tapes, flash memories, circuit configurations in Field Programmable Gate Arrays or other semiconductor devices, or other tangible computer storage medium) encoded with one or more programs that, when executed on one or more computers or other processors, perform methods that implement the various embodiments of the invention discussed above. As is apparent from the foregoing examples, a computer readable storage medium may retain information for a sufficient time to provide computer-executable instructions in a non-transitory form. Such a computer readable storage medium or media can be transportable, such that the program or programs stored thereon can be loaded onto one or more different computers or other processors to implement various aspects of the present invention as discussed above. As used herein, the term "computer-readable storage medium" encompasses only a computer-readable medium that can be considered to be a manufacture (i.e., article of manufacture) or a machine. Alternatively or additionally, the invention may be embodied as a computer readable medium other than a computer-readable storage medium, such as a propagating signal.

The terms "program" or "software" are used herein in a generic sense to refer to any type of computer code or set of computer-executable instructions that can be employed to program a computer or other processor to implement various aspects of the present invention as discussed above. Additionally, it should be appreciated that according to one aspect of this embodiment, one or more computer programs that when executed perform methods of the present invention need not reside on a single computer or processor, but may be distributed in a modular fashion amongst a number of different computers or processors to implement various aspects of the present invention.

Computer-executable instructions may be in many forms, such as program modules, executed by one or more computers or other devices. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or

implement particular abstract data types. Typically the functionality of the program modules may be combined or distributed as desired in various embodiments.

Also, data structures may be stored in computer-readable media in any suitable form. For simplicity of illustration, data structures may be shown to have fields that are related through location in the data structure. Such relationships may likewise be achieved by assigning storage for the fields with locations in a computer-readable medium that conveys relationship between the fields. However, any suitable mechanism may be used to establish a relationship between information in fields of a data structure, including through the use of pointers, tags, or other mechanisms that establish relationship between data elements.

Various aspects of the present invention may be used alone, in combination, or in a variety of arrangements not specifically discussed in the embodiments described in the foregoing and is therefore not limited in its application to the details and arrangement of components set forth in the foregoing description or illustrated in the drawings. For example, aspects described in one embodiment may be combined in any manner with aspects described in other embodiments.

Also, the invention may be embodied as a method, of which an example has been provided. The acts performed as part of the method may be ordered in any suitable way. Accordingly, embodiments may be constructed in which acts are performed in an order different than illustrated, which may include performing some acts simultaneously, even though shown as sequential acts in illustrative embodiments.

Use of ordinal terms such as “first,” “second,” “third,” etc., in the claims to modify a claim element does not by itself connote any priority, precedence, or order of one claim element over another or the temporal order in which acts of a method are performed, but are used merely as labels to distinguish one claim element having a certain name from another element having a same name (but for use of the ordinal term) to distinguish the claim elements.

Also, the phraseology and terminology used herein is for the purpose of description and should not be regarded as limiting. The use of “including,” “comprising,” or “having,” “containing,” “involving,” and variations thereof herein, is meant to encompass the items listed thereafter and equivalents thereof as well as additional items.

In the attached claims, various elements are recited in different claims. However, the claimed elements, even if recited in separate claims, may be used together in any suitable combination.

Claims

What is claimed is:

1. A network system for increasing data throughput and decreasing transmission delay along a data link from a source node to a sink node via a relay node, the network system comprising:

a first node configured to:

predict at least one future video encoding characteristic of a video source over a finite number of future frames,

encode a plurality of video data packets using rateless coding based on the predicted at least one future video encoding characteristic by adaptively adjusting at least one parameter of the rateless coding based on the predicted at least one future video encoding characteristic, and

transmit the plurality of video data packets,

wherein the video source is a video source of the plurality of video data packets.

2. The network system of claim 1, wherein:

the rateless coding comprises fountain coding,

the first node is configured to predict the at least one future video encoding characteristic based on at least one video encoding characteristic of the video source, and

adaptively adjusting the at least one parameter of the rateless coding comprises:

using model predictive control by adaptively selecting, based on the predicted at least one future video encoding characteristic, a first length of a first data window and a second length of a second data window of a plurality of overlapping sliding data windows, the first length comprising a first number of frames that the first data window can accommodate, the second length comprising a second number of frames that the second data window can accommodate, and

segmenting data from frames of the video source into the plurality of video data packets based on at least the first length and the second length.

3. The network system of claim 2, wherein:

the at least one future video encoding characteristic comprises a future video bit rate of the video source, the future video bit rate being variable.

4. The network system of claim 2, wherein:

the plurality of overlapping sliding data windows are non-homogeneous and collectively have more than one length and/or more than one sampling distribution.

5. The network system of claim 2, wherein:

adaptively adjusting the at least one parameter of the rateless coding comprises:

adaptively adjusting a first sampling distribution for the first data window and a second sampling distribution for the second data window.

6. The network system of claim 2, wherein:

the at least one video encoding characteristic comprises:

a past frame rate of the video source,

a number of frames in a group of pictures of the video source, and/or

a past video bit rate of the video source, the past video bit rate being variable.

7. The network system of claim 1, wherein:

the data link is at least partially wireless.

8. The network system of claim 1, wherein at least one video data packet of the plurality of video data packets comprises at least 100 bits.

9. The network system of claim 1, wherein:

the first node comprises a streaming video server streaming live video content.

10. The network system of claim 9, wherein:

the first node is configured to transmit the plurality of video data packets to at least one second node, the at least one second node comprising:

a sink node configured to receive one or more of the plurality of video data packets from the source node via at least one relay node, or

at least one relay node configured to receive at least one of the plurality of video data packets from the source node.

11. At least one computer-readable storage medium encoded with executable instructions that, when executed by at least one processor, cause the at least one processor to perform a method for transmitting a video stream over a data link from a source node to a sink node via a relay node, the method comprising:

predicting at least one future video encoding characteristic of a video source over a finite number of future frames;

encoding a plurality of video data packets using rateless coding based on the predicted at least one future video encoding characteristic by adaptively adjusting at least one parameter of the rateless coding based on the predicted at least one future video encoding characteristic; and

transmitting the plurality of video data packets,

wherein the video source is a video source of the plurality of video data packets.

12. The at least one computer-readable storage medium of claim 11, wherein:

the predicting is based on at least one video encoding characteristic of the video source, and

adaptively adjusting the at least one parameter of the rateless coding comprises:

using model predictive control by adaptively selecting, based on the predicted at least one future video encoding characteristic, a first length of a first data window and a second length of a second data window of a plurality of overlapping sliding data windows, the first length comprising a first number of frames that the first data window can accommodate, the second length comprising a second number of frames that the second data window can accommodate, and

segmenting data from frames of the video source into the plurality of video data packets based on at least the first length and the second length.

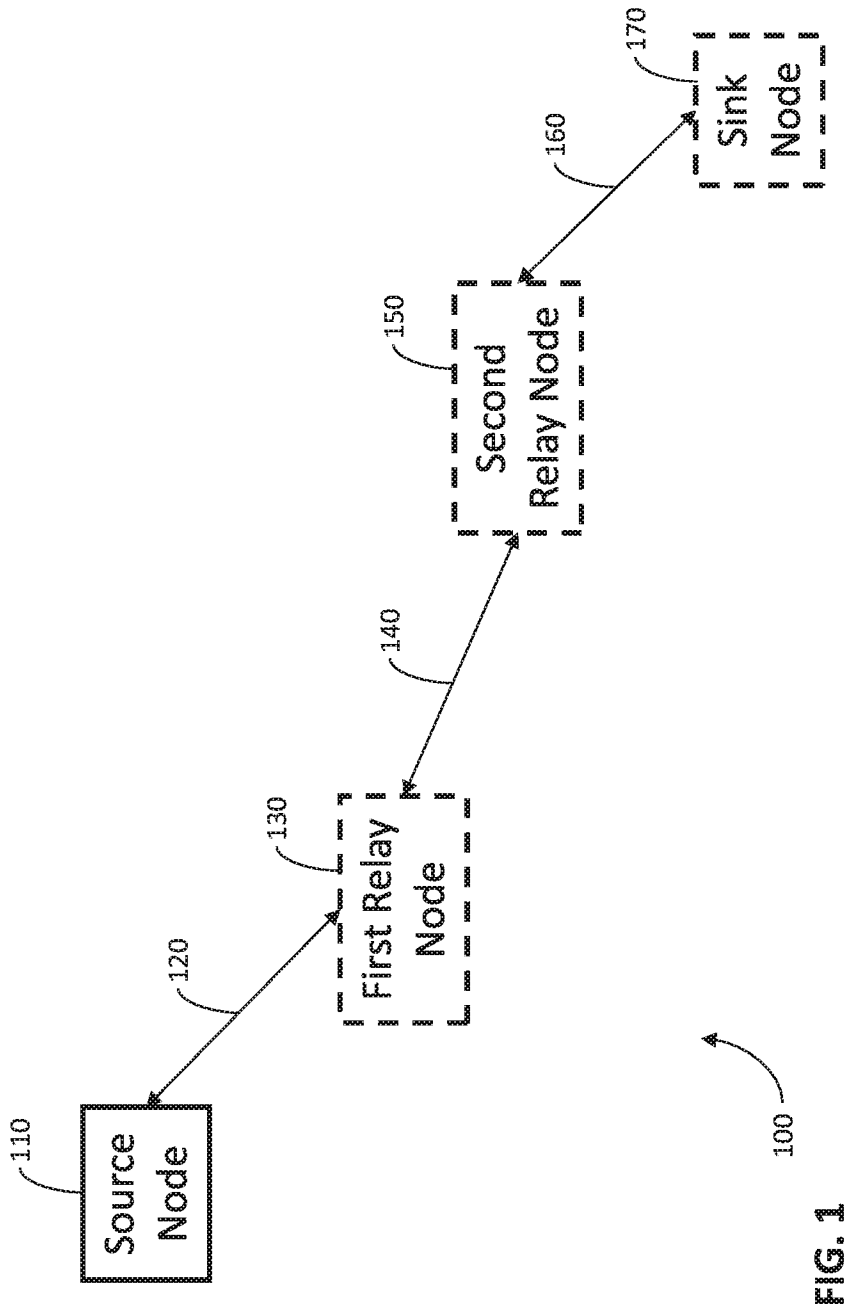
Source Node
110
120
First Relay Node
130
140
Second Relay Node
150
160
Sink Node
170
100

FIG. 1

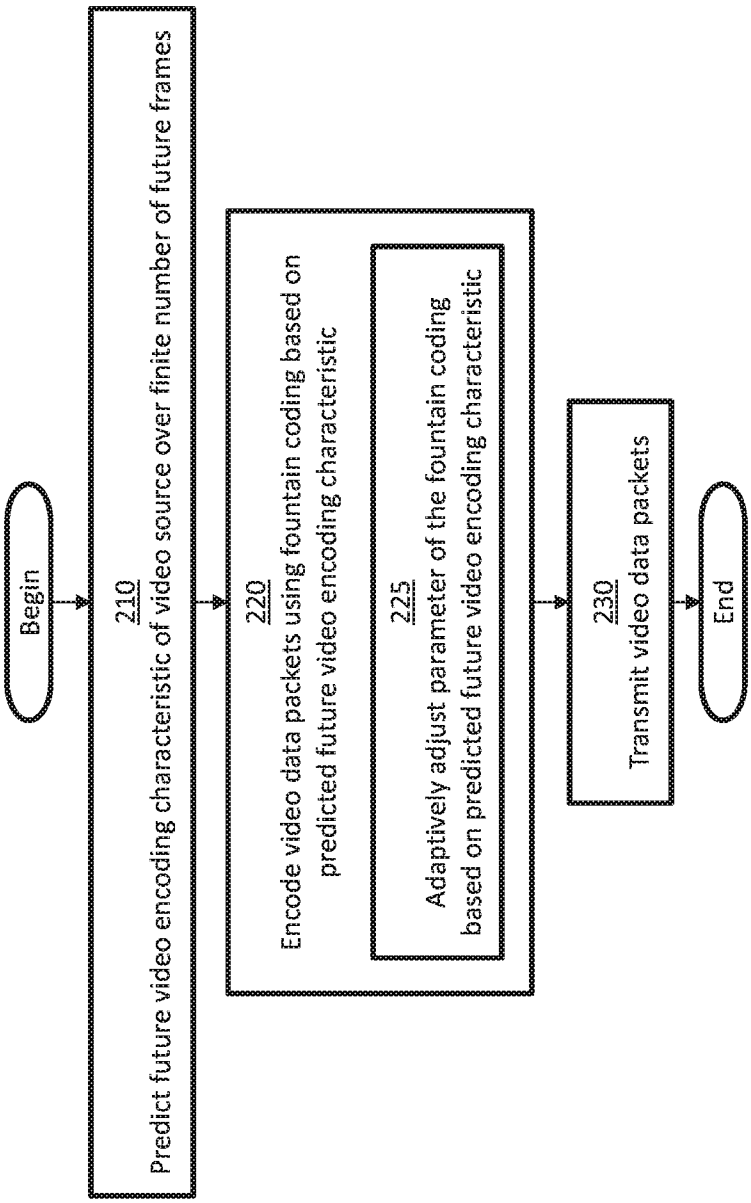
Begin
210
Predict future video encoding characteristic of video source over finite number of future frames
220
Encode video data packets using fountain coding based on predicted future video encoding characteristic
225
Adaptively adjust parameter of the fountain coding based on predicted future video encoding characteristic
230
Transmit video data packets
End

FIG. 2

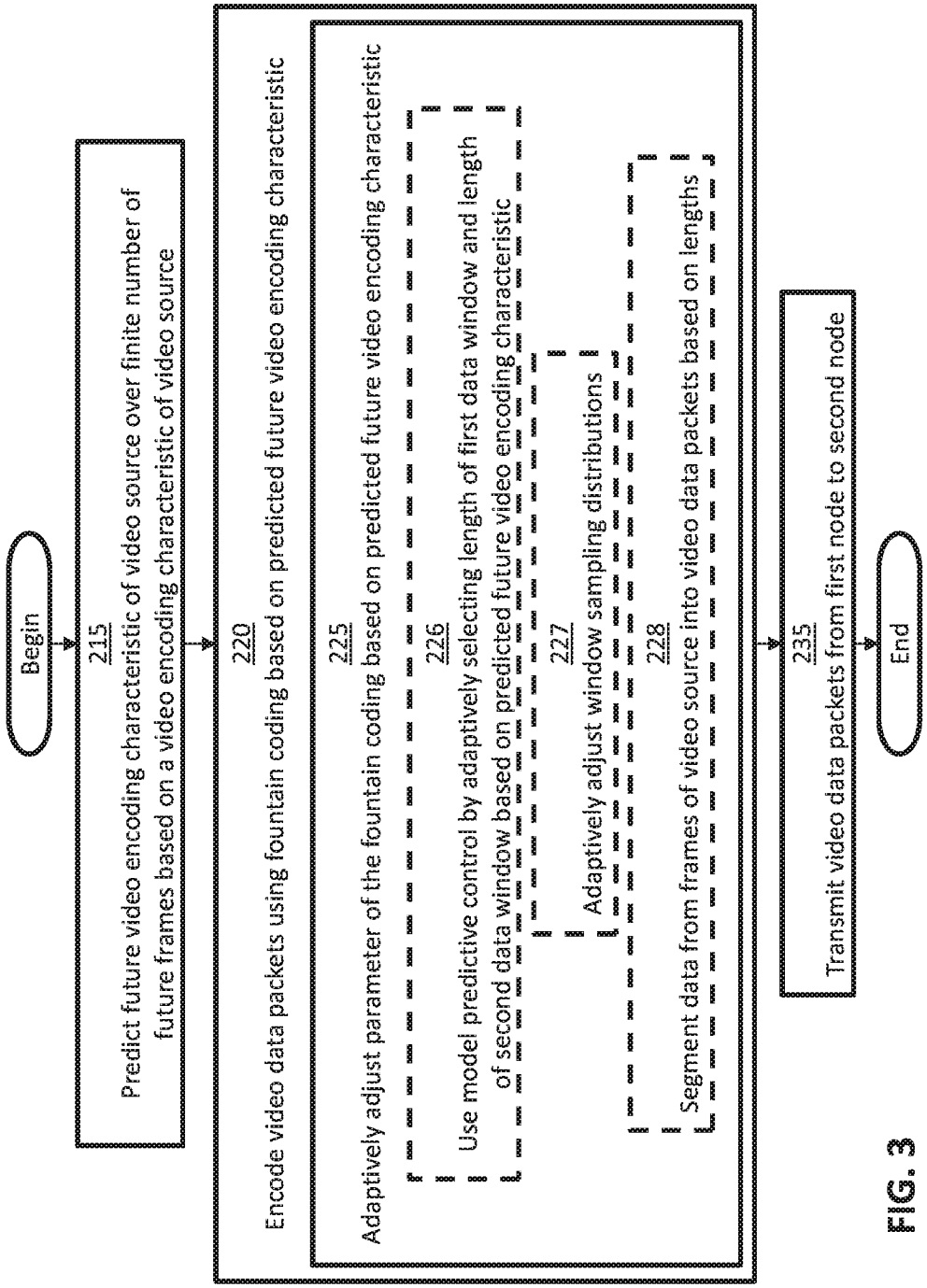
Begin
215
Predict future video encoding characteristic of video source over finite number of future frames based on a video encoding characteristic of video source
220
Encode video data packets using fountain coding based on predicted future video encoding characteristic
225
Adaptively adjust parameter of the fountain coding based on predicted future video encoding characteristic
226
Use model predictive control by adaptively selecting length of first data window and length of second data window based on predicted future video encoding characteristic
227
Adaptively adjust window sampling distributions
228
Segment data from frames of video source into video data packets based on lengths
235
Transmit video data packets from first node to second node
End

FIG. 3

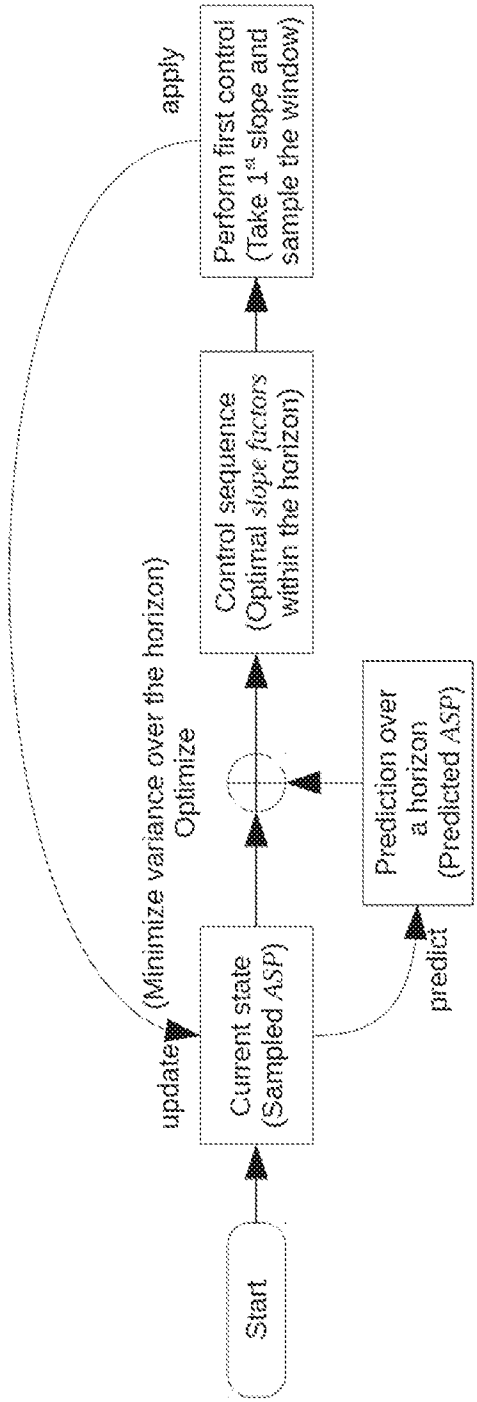
Start
Current state
(Sampled ASP)
Prediction over
a horizon
(Predicted ASP)
predict
update
(Minimize variance over the horizon)
Optimize
Control sequence
(Optimal slope factors
within the horizon)
Perform first control
(Take 1st slope and
sample the window)
apply

FIG. 4

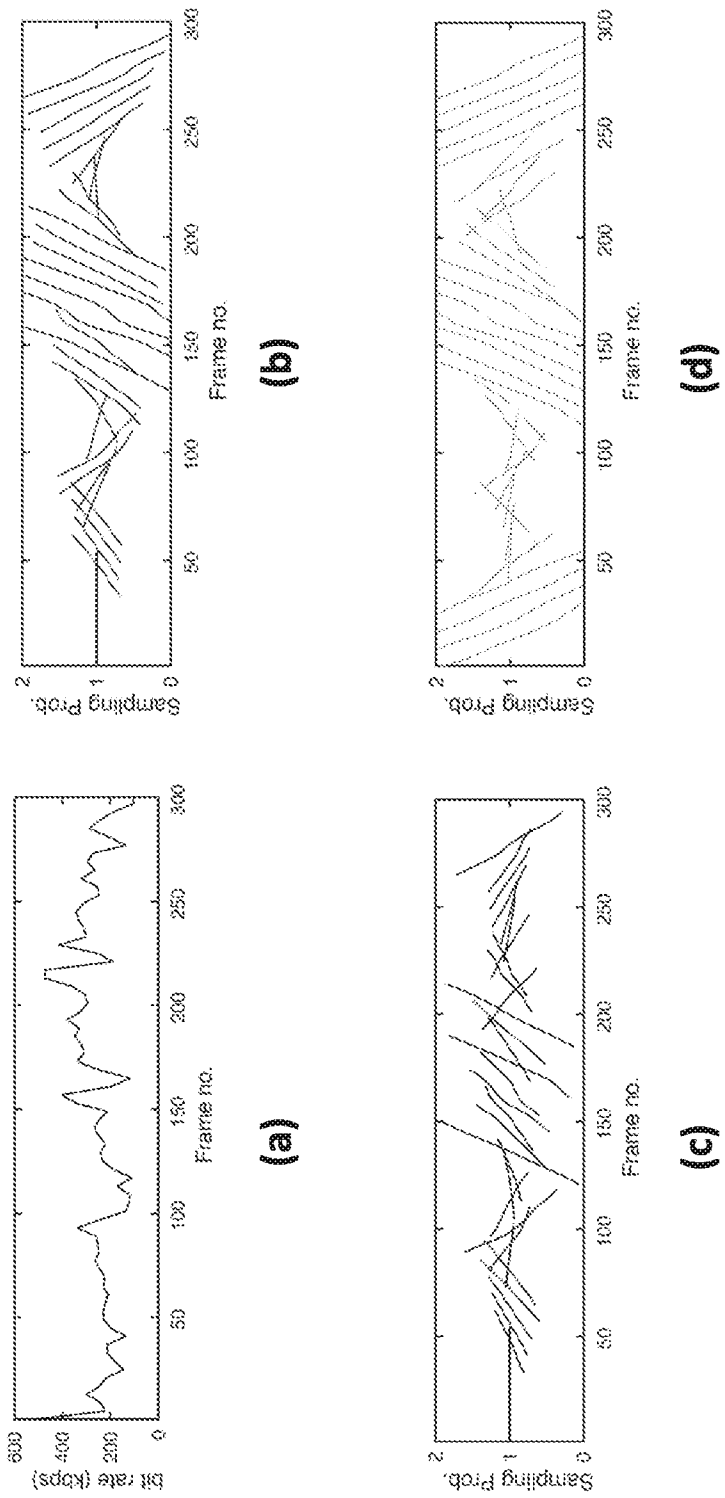
bit rate (kbps)
Frame no.
(a)
Sampling Prob.
Frame no.
(b)
Sampling Prob.
Frame no.
(c)
Sampling Prob.
Frame no.
(d)

FIG. 5

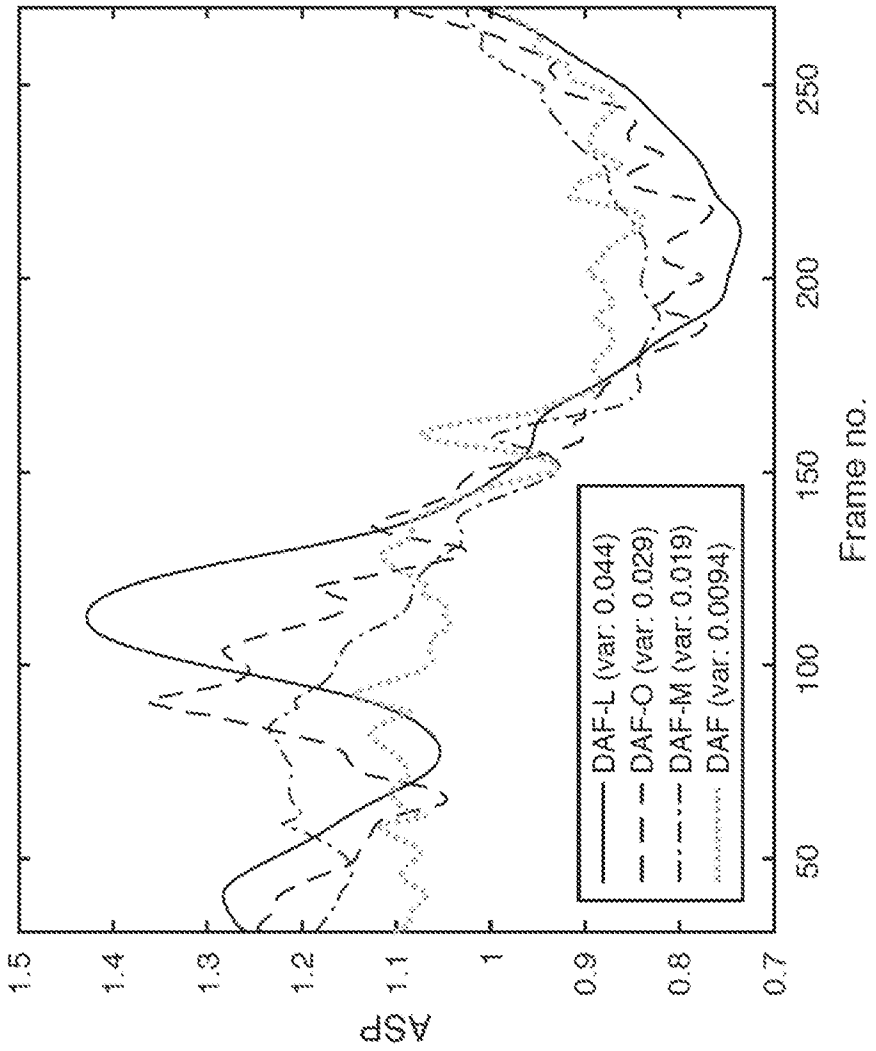
ASP
Frame no.
DAF-L (var: 0.044)
DAF-O (var: 0.029)
DAF-M (var: 0.019)
DAF (var: 0.0094)
1.5
1.4
1.3
1.2
1.1
1
0.9
0.8
0.7
50
100
150
200
250

FIG. 6

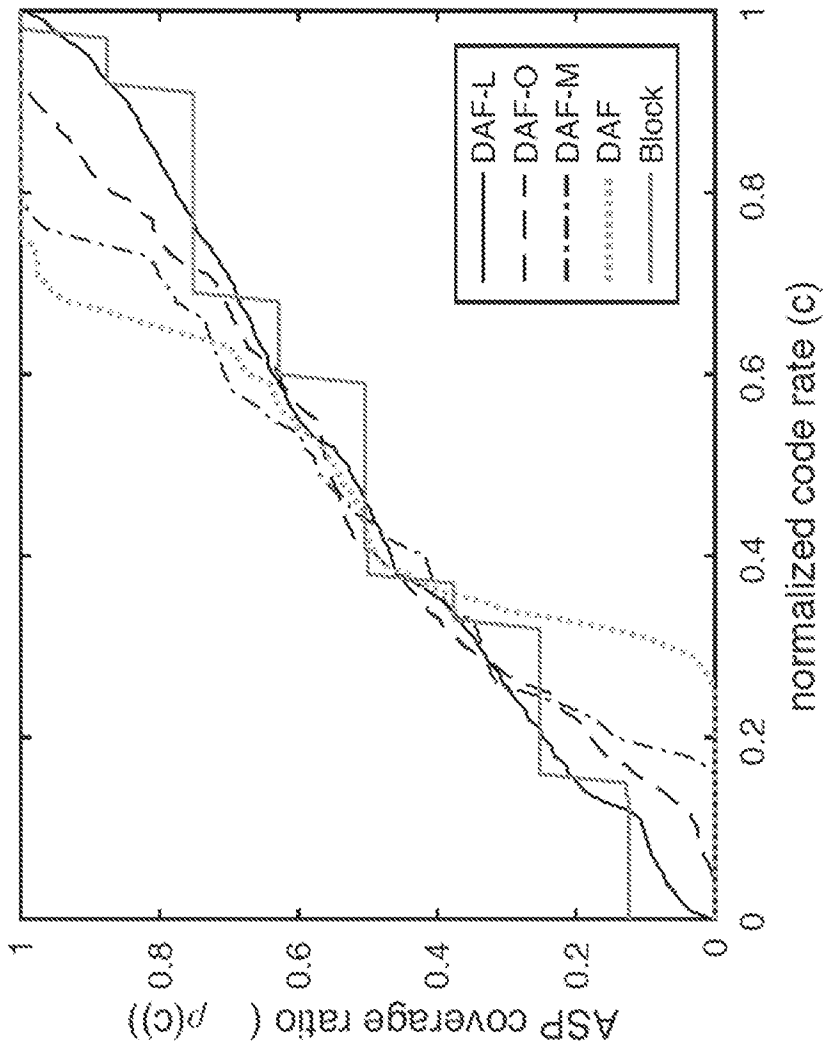
DAF-L
DAF-O
DAF-M
DAF
Block
normalized code rate (c)
ASP coverage ratio (ρ(c))
0
0.2
0.4
0.6
0.8
1

FIG. 7

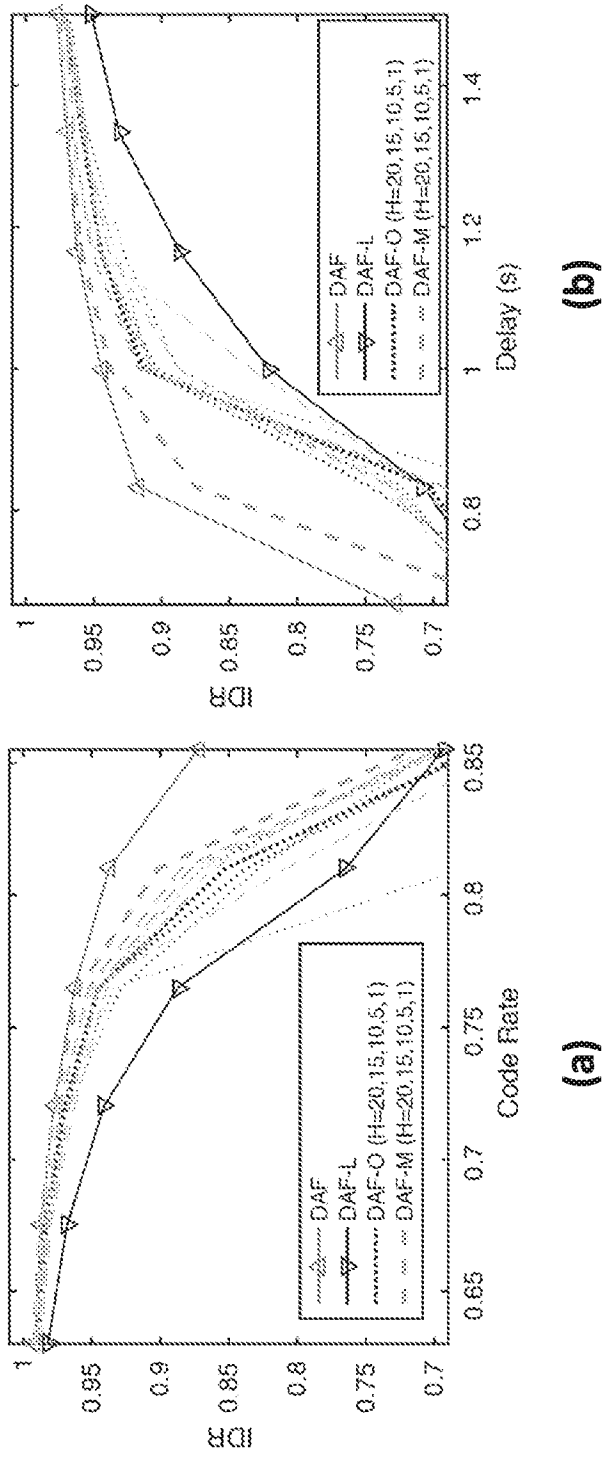
PDR
Code Rate
Delay (s)
DAF
DAF-L
DAF-O (H=20,15,10,5,1)
DAF-M (H=20,15,10,5,1)
(a)
(b)

FIG. 8

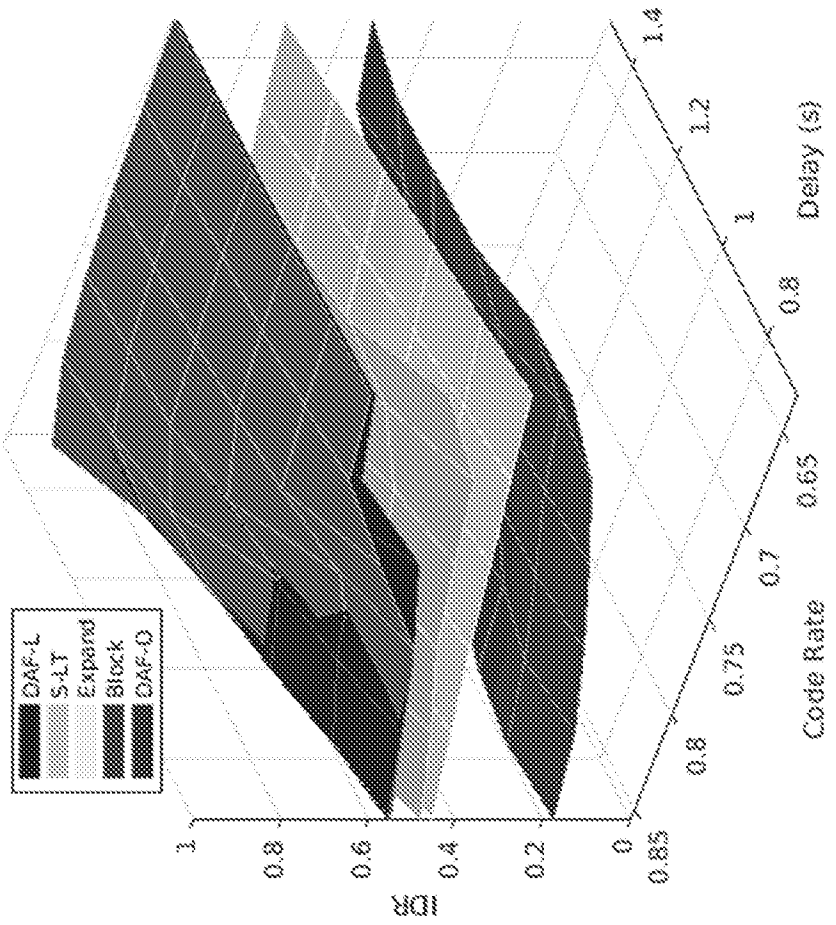
DAF-L
S-LT
Expand
Block
DAF-O
IDR
1
0.8
0.6
0.4
0.2
0
0.85
0.8
0.75
0.7
0.65
Code Rate
0.8
1
1.2
1.4
Delay (s)

FIG. 9

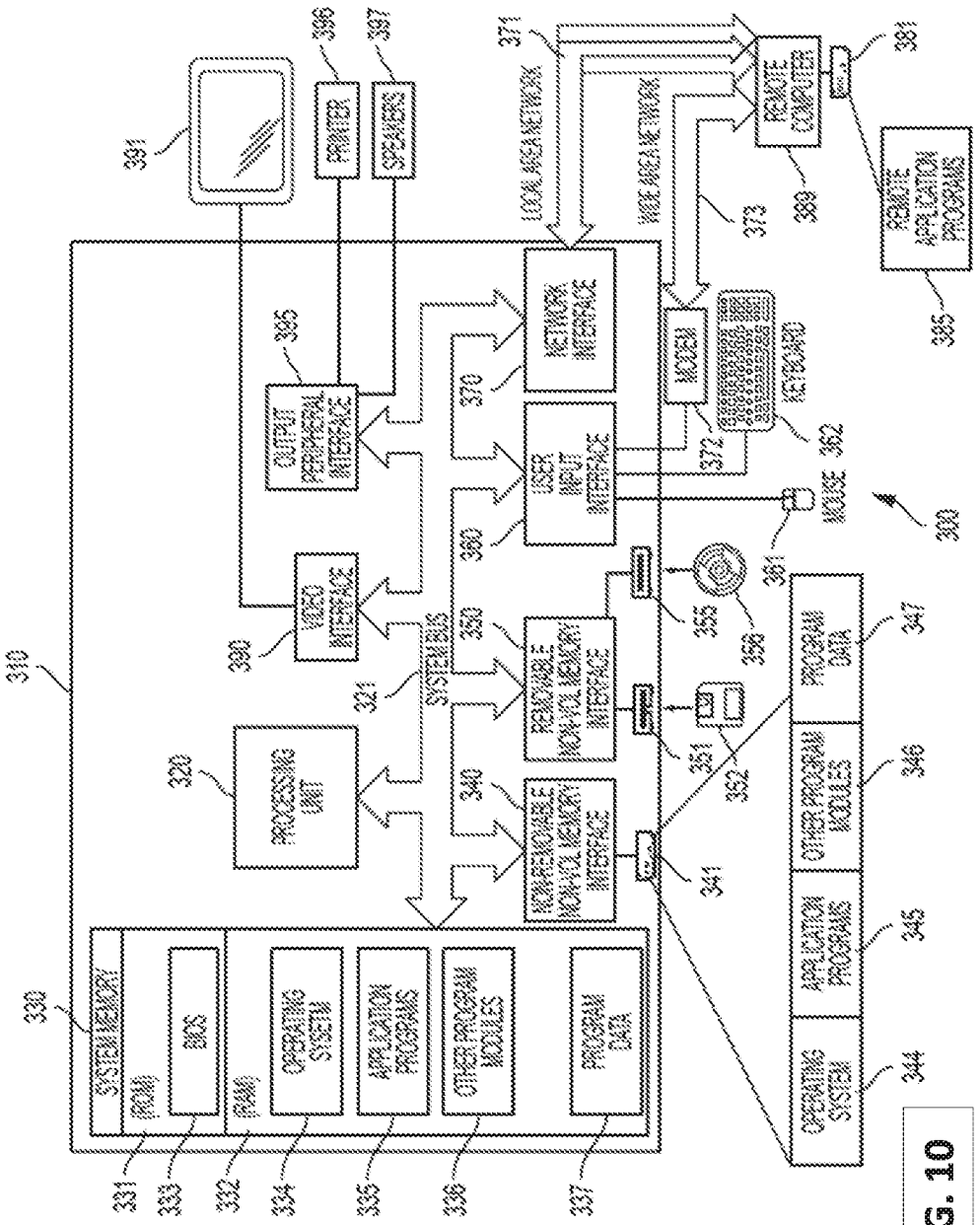
310
330
SYSTEM MEMORY
ROM
RAM
BIOS
OPERATING SYSTEM
APPLICATION PROGRAMS
OTHER PROGRAM MODULES
PROGRAM DATA
331
333
332
334
335
336
337
320
PROCESSING UNIT
321
SYSTEM BUS
340
NON-REMOVABLE NON-VOLATILE MEMORY INTERFACE
341
350
REMOVABLE NON-VOLATILE MEMORY INTERFACE
351
352
355
356
360
USER INPUT INTERFACE
361
MOUSE
362
KEYBOARD
370
NETWORK INTERFACE
371
LOCAL AREA NETWORK
372
MODEM
373
WIDE AREA NETWORK
380
VIDEO INTERFACE
390
391
MONITOR
395
OUTPUT PERIPHERAL INTERFACE
396
PRINTER
397
SPEAKERS
381
REMOTE COMPUTER
389
REMOTE APPLICATION PROGRAMS
385
OPERATING SYSTEM
344
APPLICATION PROGRAMS
345
OTHER PROGRAM MODULES
346
PROGRAM DATA
347

FIG. 10

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US17/22715

A. CLASSIFICATION OF SUBJECT MATTER
IPC - H04N19/102; H04N19/50; H04N7/24 (2017.01)
CPC - H04N19/102; H04N19/177; H04N19/184; H04N19/50

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
See Search History document

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
See Search History document

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
See Search History document

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y -- A	US 2011/0170591 A1 (LI, Z et al.) 14 July 2011, paragraphs [0022], [0037], [0069]-[0072], [0098], [0103]-[0104]	1, 7-11 --- 2-6, 12
Y -- A	WO 2016/022982 A1 (UNIVERSITY OF FLORDIA RESEARCH FOUNDATION, INC) 11 February 2016, page 4 lines 1-5, page 7 lines 15-20	1, 7-11 --- 2-6, 12
Y	US 2013/0322287 A1 (BONTU, C et al.) 05 December 2013, paragraphs [0039], [0051]	8
A	US 2015/0179183 A1 (FRAUNHOFER-GESELLSCHAFT ZUR FOERDERUNG DER ANGEWADTEN FORSCHUNG E.V.) 25 June 2015, paragraphs [0019], [0082]	1-12

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
09 May 2017 (09.05.2017)	12 JUN 2017
Name and mailing address of the ISA/ Mail Stop PCT, Attn: ISA/US, Commissioner for Patents P.O. Box 1450, Alexandria, Virginia 22313-1450 Facsimile No. 571-273-8300	Authorized officer Shane Thomas PCT Helpdesk: 571-272-4300 PCT OSP: 571-272-7774

Form PCT/ISA/210 (second sheet) (January 2015)