Journal of King Saud University – Computer and Information Sciences 32 (2020) 730–740

Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

Formal verification of a peer-to-peer streaming protocol

Oluwafolake E. Ojo [a,*], Ayodeji O. Oluwatope [b], Suraju O. Ajadi [c]

[a] Department of Computer Science, Federal University of Agriculture, Abeokuta, Nigeria
[b] Department of Computer Science and Engineering, Obafemi Awolowo University, Ile-Ife, Nigeria
[c] Department of Mathematics, Obafemi Awolowo University, Ile-Ife, Nigeria

ARTICLE INFO

Article history:
Received 3 May 2018
Revised 4 August 2018
Accepted 15 August 2018
Available online 25 August 2018

Keywords:
Peer-to-peer networks
Video streaming and temporal logic

ABSTRACT

Peer (P2P) networks have emerged as an efficient and affordable means of transmitting videos to numerous end-users via the Internet. The dynamic and heterogeneous nature of P2P streaming systems (P2PSS) makes testing, analyzing and verification a cumbersome task. However, formal methods offer efficient approaches to rigorously analyze and verify P2PSS. This paper demonstrates the use of formal verification techniques for analyzing the behavioral properties of P2PSS. We use temporal logics to analyze whether all the possible behaviors within the P2P streaming systems conform to the defined specifications. Specifically, we apply model checking to check the consistency, completeness and certainty of the model if the temporal properties of the proposed system satisfies the required specifications. Furthermore, the P2PSS framework was modeled and verified using Simulink Design Verifier (SDV) in MATLAB simulation tool. The simulations results showed 100% validation for all frames and 50% validation for I-frames prioritisation. Further, the probability of a peer capable of forwarding frames while receiving is at most 0.5.

1. Introduction

The Internet streaming technology suffices as an excellent substitute for broadcasting cable and satellite television contents in a more flexible manner (Xiao and Ye, 2008) and the fusion of P2P approach into this technology has emerged as a propitious tool for video streaming applications (Ullah et al., 2012b). Diverse P2P systems have been designed for file sharing, data dissemination and audio/video streaming which occupy substantial part of global Internet traffic (Brienza et al., 2016). In P2P streaming paradigm, multiple media channels are easily transmitted to numerous users concurrently, thereby, attracting millions of viewers daily (Shen et al., 2015).

The achievement of P2PSS is hinged on peer's ability to stream without a centralized server (Wallach, 2003), by pushing complexity from the network to users (peers), which in turn relieves the burden of bandwidth cost on the servers (Alessandria et al., 2011). P2PSS leverages the cooperation among peers, guaranteeing the service of video requests with increased scalability, reduced cost and ease of deployment (Merani and Natali, 2016) without sacrificing the quality of the stream and user experience of all peers in the network (Naiem and El-Beltagy, 2016; Magharei and Rejaie, 2009). Peers in typical P2P scenarios receive contents and easily distribute it to neighboring peers (Ullah et al., 2013). However, intermittent joining of peers or flash crowd situations (Ullah et al., 2012a; Chen et al., 2014) pose a challenge related to rendering quality of service for effective live streaming services. The effectiveness of these applications depends largely on peers' behavior and cooperation (Gonçalves et al., 2016).

Testing and analyzing peers' behaviors and cooperation is another difficult and cumbersome task, due to often large scale and heterogeneous nature of the P2P systems (Sandvik and Sere, 2011). A well-known alternative to simulation and testing is the use of formal verification technique, called model checking (Clarke et al., 1999). Formal methods provide an efficient way for formal verification of P2P systems (Gomes et al., 2012). Model checking is the formal process through which a desired behavioral property (the specifications) is verified to hold for a given model via an exhaustive enumeration, either explicit or symbolic (Rozier, 2011).

In model checking, the specification is expressed in temporal logic and the system is modeled as a finite state machine (Biere et al., 1999). Temporal logics allow the specification of the system behavior in terms of logical formulas, including temporal constraints, events, and the relationships between them (Bellini et al., 2000). Failure of the model to satisfy a desired property of the system indicates either that the model does not accurately represent the behaviors of the system, or the existence of some error(s) in the system modelling (Miller et al., 2006).

* Corresponding author.
E-mail addresses: ojoeo@funaab.edu.ng (O.E. Ojo), aoluwato@oauife.edu.ng (A.O. Oluwatope), sajadi@oauife.edu.ng (S.O. Ajadi).

https://doi.org/10.1016/j.jksuci.2018.08.008
1319-1578/

In recent decades, model checking plays a key role in system analysis and verification. For example, model checking was successfully applied in hardware verification (Clarke et al., 2003), specification and verification of security models, protocols and properties (Dixon et al., 2007; Basin et al., 2011). In addition, model checking was applied to check whether temporal properties are satisfied on all the possible behaviors of robotic swarms (Dixon et al., 2012) and security testing of web-based applications in which test cases were automatically derived from counter examples found through model checking (Armando et al., 2010). Furthermore, a new approach to model checking in security-sensitive business processes was designed in Armando and Ponta (2009). This approach allows separate specification of the work flow and the associated security policy while it retains the ability to carry out a fully automatic analysis of the process.

Datta et al., 2007 presented a protocol composition logic for proving security properties of network protocols that use public and symmetric key cryptography. The analysis of genetic regulatory networks under parameter uncertainty, using discrete abstractions and model checking was given in Batt et al. (2008).

Liu et al., 2013 proved that the discrete synthesis problem for an over-approximation can be recast as a two-player temporal logic game and off-the-shelf software can be used to solve the resulting problem.

Davies, 2017 examined the basic relationship between temporal logic and languages that involve multiple stages. The integration of linear temporal logic within the framework of ontological physics-based motion was also designed, it provided robustness and autonomy for handling complex temporal goals in a more realistic way (Akbari and Rosell, 2017).

Leveraging on the application of model checking and temporal logic for various system analysis and verification, we applied model checking to a P2P streaming protocol named UStream (Ojo et al., 2015) in order to test whether the temporal properties are satisfied on all the possible behaviors within the system. Although, the dynamic analysis of UStream was presented in Ojo et al. (2017), this paper further analyzed and verified the scheme with an attempt to check for the consistency and completeness. The other parts of the paper are organized as thus: Section 2 provides background information on video coding technology. Section 3 discusses existing video streaming protocols. Section 4 presents UStream protocol overview and the model specifications for each state transition in UStream protocol. Section 5 verifies and validates the behavior of all the steps involved in UStream protocol. Section 6 analyses the generated results and Section 7 concludes this paper and discusses potential future direction.

2. Video coding technology

The main objective of most digital video coding standards is to optimize coding efficiency (Ohm et al., 2012; Barekatain et al., 2013; Barekatain et al., 2015). The most popular organizations that have predominated video compression standardization are International Telecommunications Union – Telecommunications Standardization Sector (ITU-T) and International Standardization Organisation and International Electro-technical Commission (ISO/IEC). ITU-T concentrated on telecommunication applications and has produced the H.26x standards while ISO/IEC is more focused on consumer applications and has defined the Joint Photographic Experts Group (JPEG) standards and the Moving Picture Experts Group (MPEG) standards (Golston, 2004; Sullivan, 2005). The chronology of video coding standard is given in Fig. 1.

In video compression, classifying frames of a video file into groups of pictures (GoP) is essential in achieving high compression ratio (Le et al., 2017). A video picture is usually split up into sequences of consecutive similar/closely related pictures (Haskell and Puri, 2012). MPEG-2 standard introduced three kinds of frame types: intra-coded frame (I-frame), predictive frame (P-frame) and bi-predictive frame (B-frame) (Jiang et al., 2011; Yao et al., 2017). These frame types were also extended to subsequent coding standards. Generally, the first image in a video sequence is invariably an I-frame. I-frames are coded using only intra-frame prediction and these frames are used as references for the P-frame and B-frame prediction (Zatt et al., 2010; Ghaeini et al., 2013; Ghaeini and Akbari, 2014; Ghaeini et al., 2016). I-frames are required as starting or resynchronization points. They are also used to implement fast-forward, rewind and other random access functions. An encoder will automatically insert I-frames at regular intervals or on demand if new clients are expected to join in viewing a stream (Apostolopoulos and Wee, 1999). A group of frames from one I-frame to the frame immediately preceding the next I-frame is commonly referred to as a GoP (Seeling and Reisslein, 2014).

The encoding of a P-frame is intended to reduce the temporal redundancies across frames, thus affording better compression rates (Wang and Farid, 2006). P-frames, which are more compressible than I-frames (Yao et al., 2014), use the information of previous frames (I-and P-frames) to decompress. B-frame is the most compressed frame that utilizes previous and forward frames (I- and P-frames) as reference data (Yang et al., 2016). The earlier standards such as MPEG-1 and MPEG-2 supported the classic GoP structure shown in Fig. 2; it starts with an I slice. P slices are inserted at intervals.

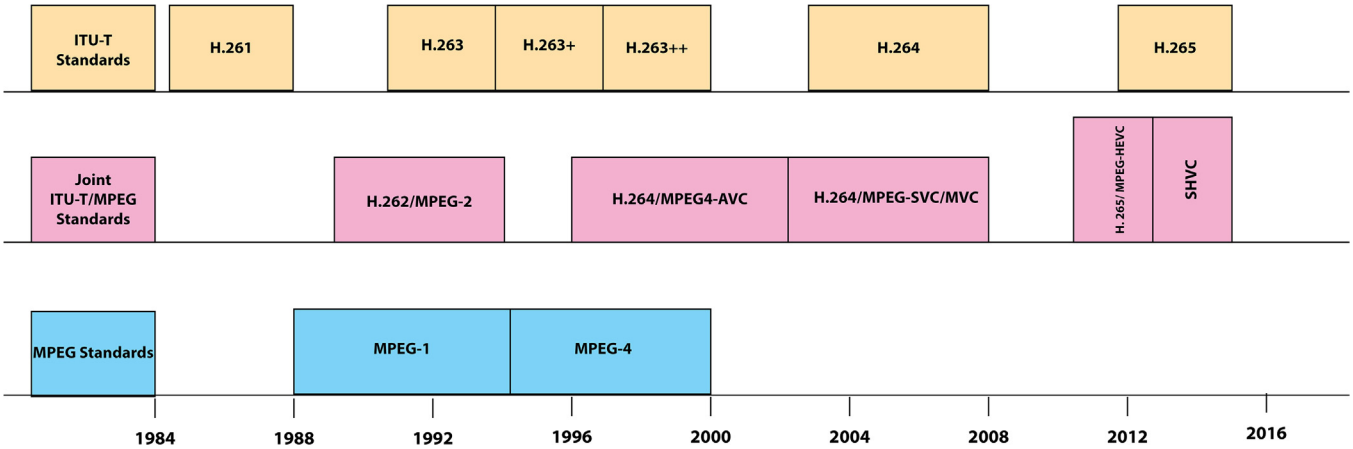

Fig. 1. Chronology of video coding standards (adapted from Golston (2004)).

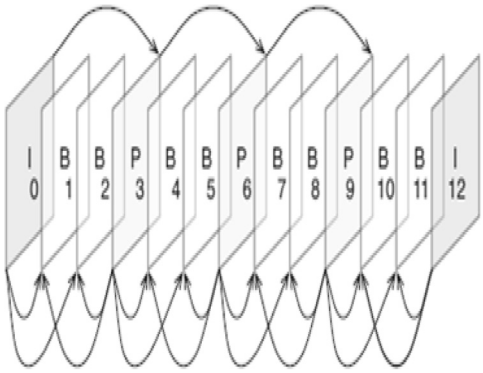

Fig. 2. Classic GoP (Richardson, 2010).

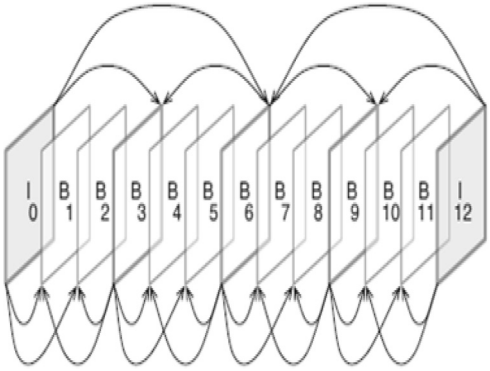

Fig. 3. Hierarchical GOP structure (Richardson, 2010).

Moreover, B slices are inserted between I and P slices. The major shortcomings of classic GoP are increased delay and larger frame storage requirements. The H.264 flexibility prediction option motivated the concept of hierarchical GoP structure as shown in Fig. 3 and extended to H.265. In hierarchical GoP structure, the GoP starts with I_0 and finishes with I_{12}. Then, slice B_6 is predicted using I_0 and I_{12} as references. $B_3$is predicted from I_0 and B_6; B_9 is predicted from B_6 andI_{12}. B_1 and B_2 are predicted from I_0 and B_3 and so on (Richardson, 2010).

3. Existing P2P video streaming protocols

Over the years, diverse scientific studies have been conducted on the P2P video streaming protocols, ranging from P2P overlay construction, data sharing strategy, adoption of advanced video coding, to the application of network coding technique and many more (Hu et al., 2011). The outcomes of previous studies have produced various widely deployed commercial P2P live streaming systems with a large number of Internet viewers. CoolStreaming is a data-driven overlay network for P2P live media streaming. The scheme ensures that every node periodically exchanges data availability information with a set of partners, and retrieves required data from one or more partners, or supplies available data to partners. A scalable membership and partnership management algorithm together with an intelligent scheduling algorithm was proposed, which enables efficient streaming for medium to high bandwidth contents with low control overhead. CoolStreaming achieves high streaming quality and minimal delay (Zhang et al., 2005).

SopCast is one of the most popular P2PTV applications that allows a user to broadcast video contents via a channel (SopCast, 2007). The SopCast Client has multiple choices of TV channels, each of which forms its own overlay. Each channel streams either live audio-video feeds, or loop-displayed movies according to a preset schedule. The viewer tunes into a channel of his choice and SopCast starts its own operations to retrieve the stream. After some seconds a player pops up and the stream can be seen (Fallica et al., 2008).

Anysee is another P2P live streaming scheme. It is an inter-overlay optimisation (IOO) scheme that adopts the mesh-based topology to construct efficient paths, using peers in different overlays(Liao et al., 2006). In the scheme, efficient mesh-based overlay was constructed; location detector-based algorithm was employed to match the overlay with the underlying physical topology. Secondly, a single overlay manager was adapted to deal with the join/leave operations of peers and the buffer manager was designed for management and scheduling of transmission of media data. The IOO improves global resource utilisation and distribute traffic to all physical links evenly (Liao et al., 2007).

R^2 is another streaming algorithm that incorporates random network coding with a randomized push algorithm. R^2 improves the performance of live streaming in terms of initial buffering delays, resilience to peer dynamics and bandwidth costs reduction on dedicated streaming servers. Also, seeds in R^2 receive feedback from downstream peers in a timely manner (Wang and Li, 2007).

Furthermore, PPLive is a free P2P IPTV application that provides over 200 channels to an average of 400,000 users on daily basis. Through measurement study conducted on PPLive, it has been proven that Internet infrastructure is capable of providing the performance requirements of IPTV at low cost and with minimal dedicated infrastructure (Hei et al., 2007).

A new framework for P2P media streaming named BEAM, was introduced in Purandare and Guha (2007). In BEAM, nodes cluster into groups, called alliances, for a symbiotic association in order to share the media content. Through empirical investigation, the researchers proved that alliance formation is an effective way to organize the peers in loosely coupled groups. The node topology formed, using alliances, generates a small world network, which exhibits efficient overlay structures in terms of path lengths between the nodes and robustness to network perturbations like churn. ToroStream, a P2P protocol for live video streaming was presented by some other researchers in Fiandrotti et al. (2012). This protocol was designed to minimize communication delay using push-based packet scheduler and an efficient feedback mechanism. Experimental results showed that the protocol achieved continuous streaming even on lossy networks, support hundreds of users with a small initial buffering and suitable for low-delay video communications.

Moreover, RapidStream, a proof-of-concept implementation of a P2P video streaming application for android compatible devices, was presented in Eittenberger et al. (2012). A multicast tree topology was created in Hammami et al. (2014) based on upload capacity and each group in a tree forms a mesh topology. Each group represents a level of upload capacity. It was assumed that the source possesses the most upload capacity and belongs to level 0 (the highest level). The multimedia contents are delivered from the source being a member of the highest level to the low levels. That allows moving the powerful peers close to the source. Through extensive simulations and evaluation, the topology achieved minimum start-up delay, the end-to-end delay and the play-back delay.

Also, a P2P protocol for live streaming networks was formulated. The scheme employs a packet integration mechanism along with a random network coding in order to increase network throughput and video quality. It is a push-pull mesh-based protocol that gives higher priority to the base video layers for transmission among peers. It was assumed that the system is a trust-based system and no data poisoning is possible. The data, however, may

get lost during the communication due to the noise or the missed deadline (Ayatollahi et al., 2018) and many more.

From the review of related work, it was discovered that P2P streaming protocols are implemented on either tree-based topology and mesh-based topology with efforts to control delay and flash crowd over the network. It was observed that the tree-based topologies are represented in form of a structured hierarchy, while the mesh-based topology presented an unstructured hierarchy. The structured system ensures orderliness in the network because peers join and leave the network in a fashionable manner as defined by the overlay topology but the major shortcoming of structured topology is the dependability of the children peers on the equivalent parent peers. This implies that a failure at the parent peer connotes a failure to all the attached children's peers which makes it unstable in dynamic environments and large-scale networks.

On the other hand, an unstructured system allows peers to join and leave the network without policy enforcement, and distribute traffic to all peers evenly. In a situation where any of the parent peers fails, the children peer can directly feed from the next available peer. Unstructured topology addresses the challenge of structured topology with its ability to disseminate chunks directly to the neighbouring peers though it is liable to flash crowd and high churn situation due to unplanned interruptions. The mixture of the tree-based and the mesh-based topology is called hybrid – combining the advantages of both tree-based and mesh-based topology to improve performance of the peers in the network (Hammami et al., 2014). High churn, accommodating unreliable peers, starvation of peers and high packet loss are the major drawbacks deduced from the existing work. Hence, an ultra-metric spanning overlay topology for P2PSS (UStream Protocol) was proposed by Ojo et al. (2015).

4. UStream protocol overview

UStream is a three-layer video streaming protocol which consists of adaptation, scheduling and topology layers, as fully described in Ojo et al. (2015). Mapping UStream protocol to the standard transmission control protocol and Internet protocol (TCP/IP), raw videos are sent directly to the adaptation layer which ensures that video frames are encoded/decoded. The adaptation layer of UStream protocol is located at the application layer of TCP/IP model. The encoded video frames move from the application layer to the transport layer. In addition to the default functions of the transport layer, UStream's transport layer is enhanced with a frame scheduler referred to as scheduling layer. Therein, the scheduled frames are sent to the Internet layer (another addition) referred to as the topology layer in the UStream protocol. Specifically, a P2P overlay structure was designed for the topology layer that is located at the Internet layer of the TCP/IP model.

4.1. UStream model checking and assumptions

The model checking of the various modules in the UStream protocol was tested using boolean logic; that is, true (¬ Error) or false (Error) as shown in Fig. 4. Starting with the capture module, a simple true (¬ Error) or false (Error) check is performed. In the case of successful video capturing, the model checking process returns a true value and switches to the next stage (Encode module), otherwise the process returns false value (which means the video data will be re-captured). This process of true (¬ Error) or false(Error) checking was repeated through other modules to the playout module. The model checking can best be described as a single direction checking strategy. However, assume that an error occurs in the course of checking module(n), model checking process returns a false value and transits to module(n-1).

The Schedule and Overlay Network modules in Fig. 4 are the main contributions of this research. Sections 4.2 and 4.2 present a detailed model checking for schedule and overlay network modules using computation tree logic (CTL) to explicitly test all the possible behaviors of the states therein. The definitions for CLT operators is given in Table 1.

Table 1
Computation tree logic operators.

Symbols	Meaning
AG	(everywhere – along all paths)
EF	(everywhere – along some path)
AF	(somewhere – along all paths)
AX	(all successors)
A [ψ_1 U ψ_2]	(until – along all paths)
¬	negation
∧	conjunction
∨	disjunction

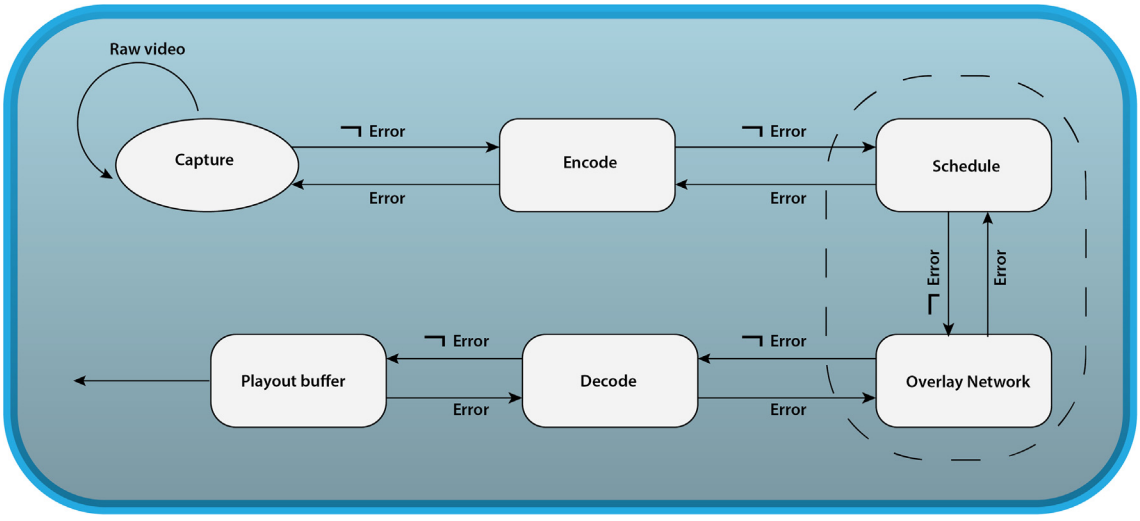

Fig. 4. UStream Protocol Model Checking.

4.2. Specification for the schedule module

The schedule module representing the scheduling layer of the UStream protocol implements a congestion control scheme which is a hybrid of weight fair queuing and leaky bucket techniques. It consists of frame classifier, time controller and traffic shaper. Frame classifier groups video frames into frame types and attach weights to these frames [I-, P- or B-frames]. The time controller allocates time slots to frames on the basis of prioritization to mitigate unnecessary delay, thereafter sends such classified frames to the traffic shaper. Finally, the traffic shaper admits the prioritized frames and attempts to regulate frame rates with the objective of achieving a steady flow of frames using the leaky bucket technique. In our modelling, we assumed that the schedule module employs the hierarchical GoP structure with priority given to I-frames. Table 2 presents definition of terms used for specification. The following casespresent all possible instances and behaviors of the schedule module.

Case 1. A classifier is embedded at the input phase of the schedule module which accepts video frames and classified based on the specified frame type. Here, the aim is to check whether all frames within the schedule module along all path are classified or not. The specification is defined as "If frames arrived, then the frames are classified" and represented as: *AG (Arrived → Classify).*

Case 2. As discussed earlier, I-frame is an important frame in the GoP when compared with other frame types. Based on the functions offered by I-frame, another crucial procedure in the schedule module is prioritizing I-frame. Therefore, whenever an I-frame is encountered, it is moved immediately to the next phase without delay. Hence, we defined the specification for this instance as *"If classify is true and IFrame is true in any subsequent state, then priority will eventually become true until IFrame is false*" and it is represented as:

$$AG\ (Classify \wedge IFrames) \rightarrow A(\neg\ IFrames\ U\ Priority)).$$

Case 3. The system checks if the time controller functions as designed by forwarding frames to the traffic shaper. The specification is defined as: *"Whenever the time controller is correctly activated, it eventually sends frames to traffic shaper*" and it is symbolized as follows:

$$AG\ ((Start\text{-}Tc \wedge \neg\ Error) \rightarrow AF\ SendFrames\text{-}Ts).$$

Case 4. Another function of the time controller is to consistently check the classified frames buffer to prempt unneeded delay or starvation. The specification is given as *"Whenever the time controller is correctly activated, it eventually checks classified buffers regularly*" and it is written as:

$$AG\ ((Start\text{-}Tc \wedge \neg\ Error) \rightarrow AF\ Checks\text{-}ClassifiedBuffers).$$

Table 2
Terminologies in Schedule Module.

Terms	Meaning
Arrived	Arrival of frames
Classify	Classification of frames
Iframe	I-frame
Pframe	P-frame
Bframe	B-frame
Priority	Prioritize frame
TS	Traffic shaper
Tc	Time controller

Case 5. The possibility of fast-forwarding delayed frames within the classified frames buffer into the traffic shaper is explored. We defined the specification as *"Whenever a frame is delayed unnecessarily in the classified buffers, then it is moved into the traffic shaper immediately*" and represented as:

$$AG\ ((FrameDelay \rightarrow AF\ SendFrames\text{-}Ts).$$

Case 6. The system checks output of traffic shaper for a steady flow rate. The specification is given as *"Whenever the traffic shaper is correctly activated, it eventually sends frames at constant rate*" and it is symbolized as:

$$AG\ ((Start\ traffic\ shaper \wedge \neg\ Error) \rightarrow AF\ Constant).$$

Case 7. Lastly, the possibility of forwarding frames within the schedule module despite the occurrence of error anywhere was tested. The specification is defined as *"Whenever an error occurs, it is still possible to send frames*" and it is represented as:

$$AG\ (Error \rightarrow EF\ SendFrame).$$

4.3. Specification for overlay network module

The overlay network module represents the topology layer of the UStream protocol. A detailed information about the UStream topology layer is given in Ojo et al. (2015). We assumed that the overlay network module support peer-assisted content distribution (that is, peers can also receive frames directly from the streaming server). Table 3 presents terms used in the specification of the overlay network module and all probable steps attributes of the overlay module are specified as follows:

Case 1. The system is checked if active peers (that is, connected peers) within the overlay network module can either send or receive frames. The specification is defined as "If peer is active, then peer can send or receive frames" and it is represented as follows:

$$AG\ (Active \rightarrow Send \vee Receive).$$

Case 2. The possibility of active peers to send and receive frames simultaneously is tested. We defined the specification as *"If peer is active, then peer can send and receive frames*" and it is symbolized as:

$$AG\ (Active \rightarrow Send \wedge Receive).$$

Case 3. In the overlay network module, the peers at higher layer can easily exchange information with lower layer peers. For instance, the parent peers can send or receive frames to and from their respective children peers. The specification is defined as *"Whenever a* $layer_n$ *peer is correctly linked to layer* $_{n+1}$*, eventually layer* $_{n+1}$ *will send or receive frames*" and it is represented as:

Table 3
Terminologies in Overlay Network Module.

Terms	Meaning
Active	Connected pair of nodes
Send	Sending frames to another node
Receive	Receiving frames from another node
$layer_n$	Layers in UStream protocol(n = 1,2,3)
$layer_{n+1}$	Next low layer
Equaldistance	Peers receiving frames the same time

$AG\ ((layer_n\ links\ layer_{n+1} \wedge \neg\ Error) \rightarrow AF\ Send \vee Receive).$

Case 4. Similar to Case 3, the system checks if the higher level peers can send and receive frames to and from their respective lower-level peers. It is specified as *"Whenever a $layer_n$ peer is correctly linked to layer $_{n+1}$, eventually layer $_{n+1}$ will send and receive frames*" and symbolized as:

$AG\ ((layer_n\ links\ layer_{n+1} \wedge \neg\ Error) \rightarrow AF\ Send \wedge Receive).$

Case 5. We checked for the possibility of lower-level peers acting as approximate high-level peers in case any higher-level layer is inactive. The specification is defined as *"Whenever an error occurs at $layer_n$, eventually layer $_{n+1}$ becomes approximate layer $_n$*" and it is represented as:

$AG\ (Error\ layer_{\ n} \rightarrow EF\ (layer_{\ n+1} \rightarrow approximate\ layer_{\ n})).$

Case 6. The equidistant function of the overlay network module is tested in this instance i.e. possibility that peers frames rate is equal irrespective of differences in peers distances. The specification is given as *"All peers at layer $_{n+1}$ are equidistant from peer at layer $_n$*" and it is represented as:

$AG\ (peers \rightarrow AX\ (layer_{\ n+1} \rightarrow EqualDistance\ layer_{\ n}).$

5. Model verification

We implemented UStream streaming protocol with the simulink design verifier (SDV) within MATLAB R2017a (See Fig. 5). The capture module is implemented as an input video file (180 × 360, 30 fps, uncompressed) using the multimedia files block function with image signal set at one multidimensional. The encode and decode modules are implemented using motion compensation and discrete cosine transform techniques (adapted from MPEG video compression example in MATLAB). The output from the compression block is a stream representing sequence of video frames. We assumed that the video frames from the MPEG compression block feeds the schedule module for onward transmission of frames to the decode module (Decompression, MPEG) through the overlay network module. Lastly, regulated video frames are released from the playout buffer module implemented with a viewer function. As discussed earlier in Section 4, simple error check is performed on each module in the streaming protocol. To achieve model checking, model verification assertion function in MATLABis utilized. The assertion function is implemented for all the modules to ascertain the behavior of video frames at each stage, that is, if transmission is successful, the video frames move to the next phase, else, an error message is generated.

5.1. Verification of the schedule module

All the possible processes running within the schedule module are implemented using temporal logic function. Fig. 6 presents the parent schedule module as modeled in SDV environment. The model consists of input and output variables, verification modules, command windows, control system and logical flow. The input variables are: *Iframe, bframe, pframe, status, restart, arrive, corrupt, controller, shaper and frameDelay*, and the output variables are: *constant rate, classify, priority and traffic shaper*. Also, each specification of schedule module stated in Section 4.2 are represented using verification modules that are connected to the parent schedule module as shown in Fig. 6. These specifications are verified for accuracy and validity with boolean variables 0 (false), 1 (true) and values between 0 and 1 (degree of accuracy).

The output results from the verification modules were displayed using the command windows. Furthermore, the main function of the control system as depicted in Fig. 7 is to synchronize the input variables from the parent schedule module with logical flow. The control system serves as an intermediary between the parent module and its logical flow. The *restart, status, controller, shaper*

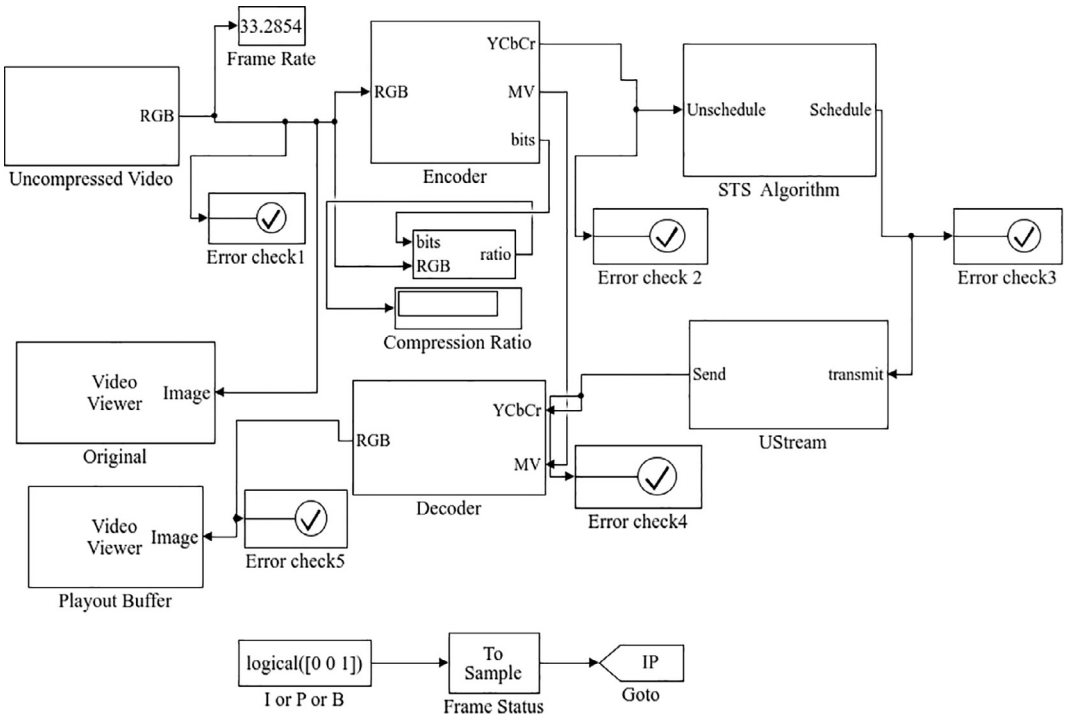

Fig. 5. UStream Protocol in Simulink.

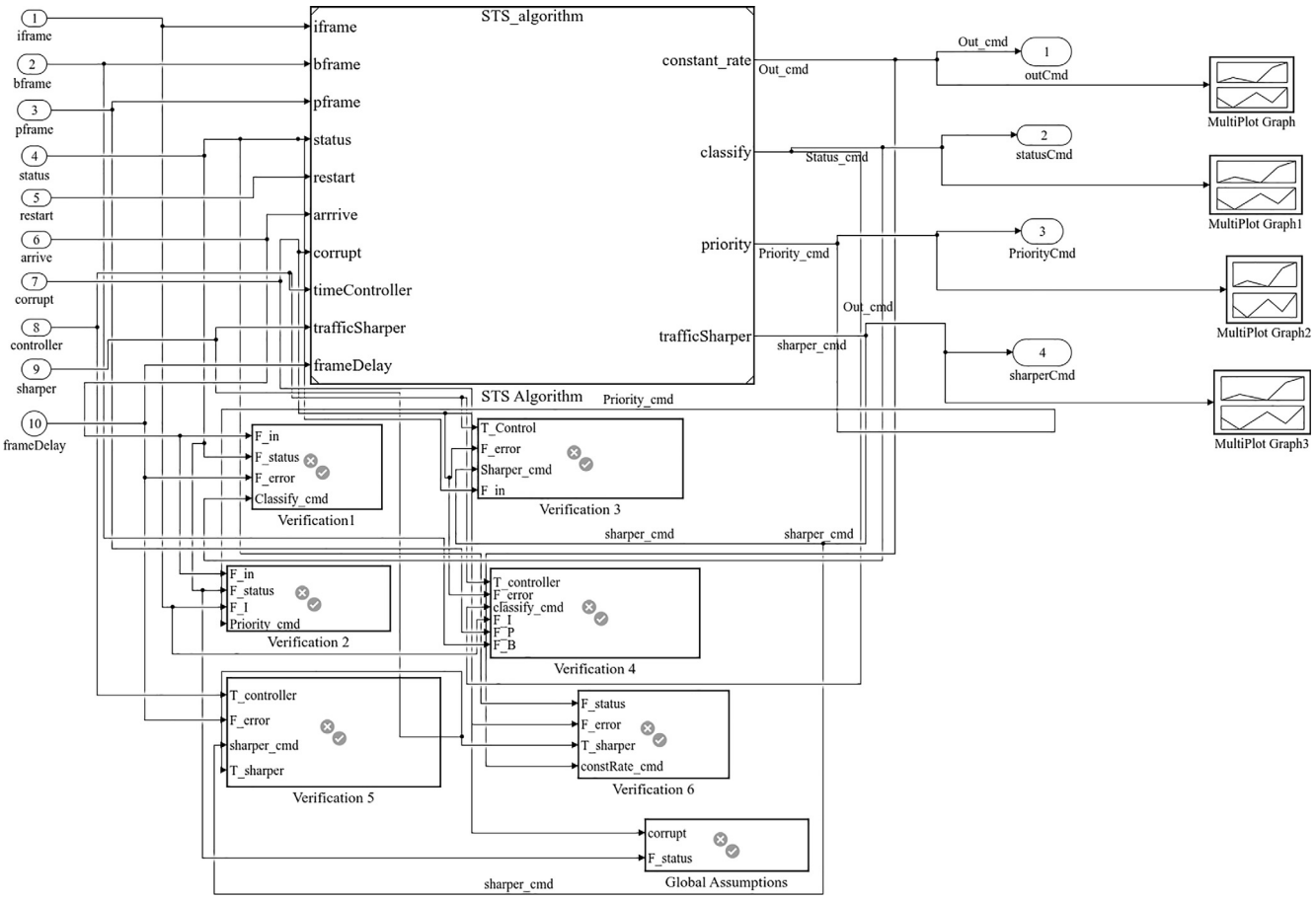

Fig. 6. Schedule Parent Module in MATLAB.

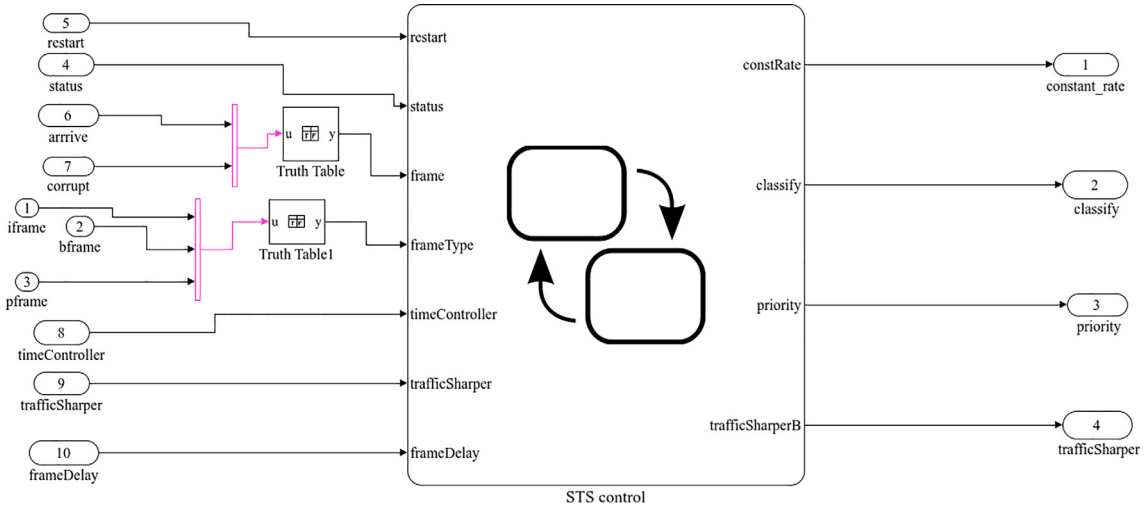

Fig. 7. Schedule Module Control System.

and *frameDelay* input variables moves to the logical flow without any process by the control system.

However, the control system checks the state of the frame before moving to the logical flow (to check if arrived frame is corrupted or not) using the truth table function in SDV. Similarly, the type of received frame is also checked for *Iframe, Pframe or Bframe*. Therefore, the input variables (arrive and corrupt) are processed by the control system to produce a single output tagged as *"frame" (arrive or corrupt)*. In addition, the control system performs logical operation on the input variables *Iframe, Pframe, Bframe* which in turn produce output "*frameType*" (*Iframe or Pframe or Bframe*) as depicted in Fig. 7. The logical flow serves as the engine of the schedule module, It describes the state flow sequence of schedule module. It accepts frames from the control system, classifies frames using prioritization scheme, put classified frames to the traffic shaper using the time controller (which also ensures that frames are not delayed unnecessarily) and ensures constant flow of frames from the traffic shaper.

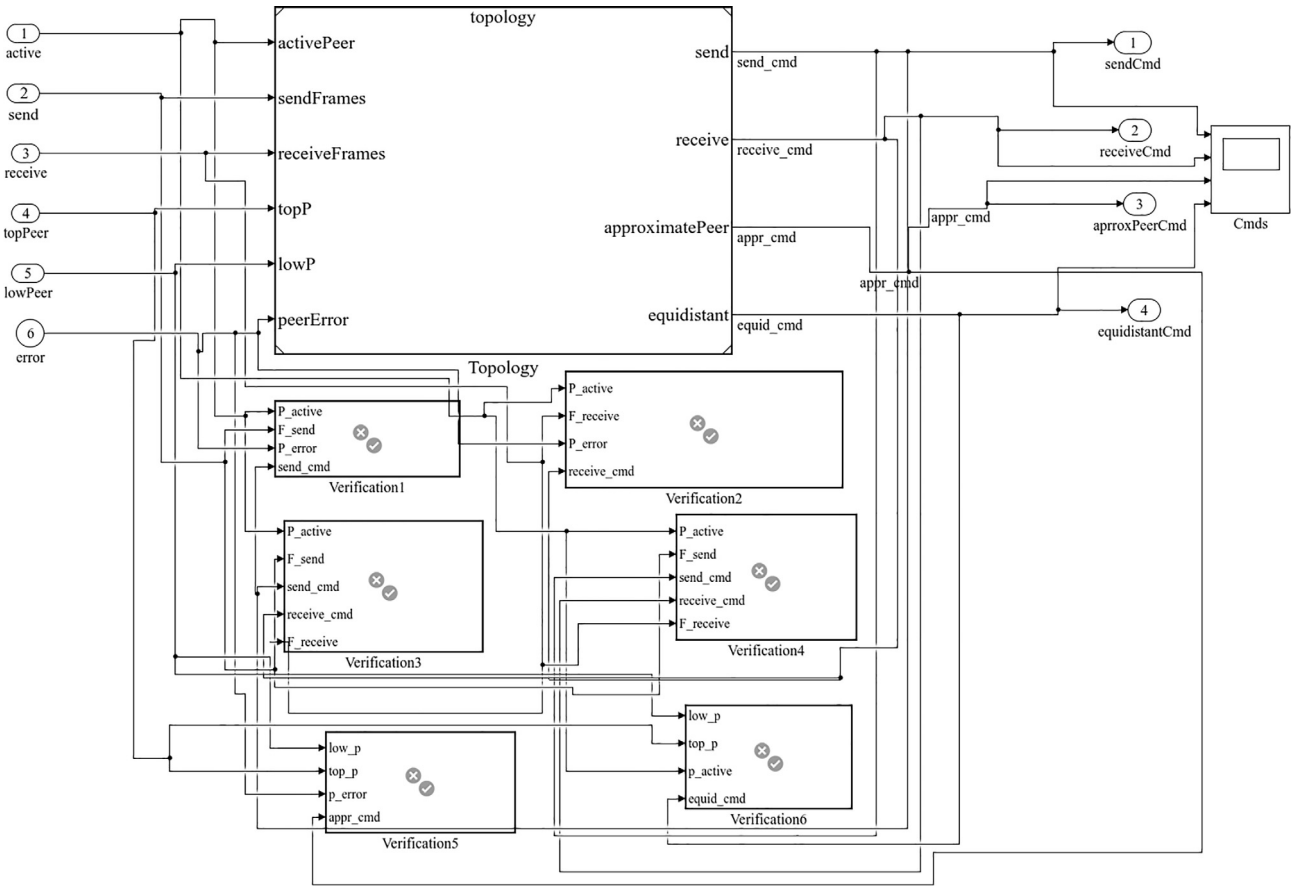

Fig. 8. Overlay-Network Parent Module in MATLAB.

5.2. Verification of the network module

The overlay module is implemented and verified in SDV environment using temporal logic function. The overlay-network parent module, as depicted in Fig. 8 consists of input and output variables, verification modules, control system, logical flow and command windows. The input variables defined for overlay-network module are: *active, send, receive, top peer, low peer and error*. The output variables specified are: *send, receive, approximate and equidistant*. Furthermore, each specification presented in Section 4.3 are implemented in the verification modules to test the consistency and accuracy of the system. The control system, as depicted in Fig. 9, serves as a middle-ware between the overlay-network parent module and logical flow. The *active* and *error* input variables are reprocessed by the control system using the truth table component (i.e., driven by boolean values (0,1)) which produces an output value *"peer" (i.e., active or error)*.

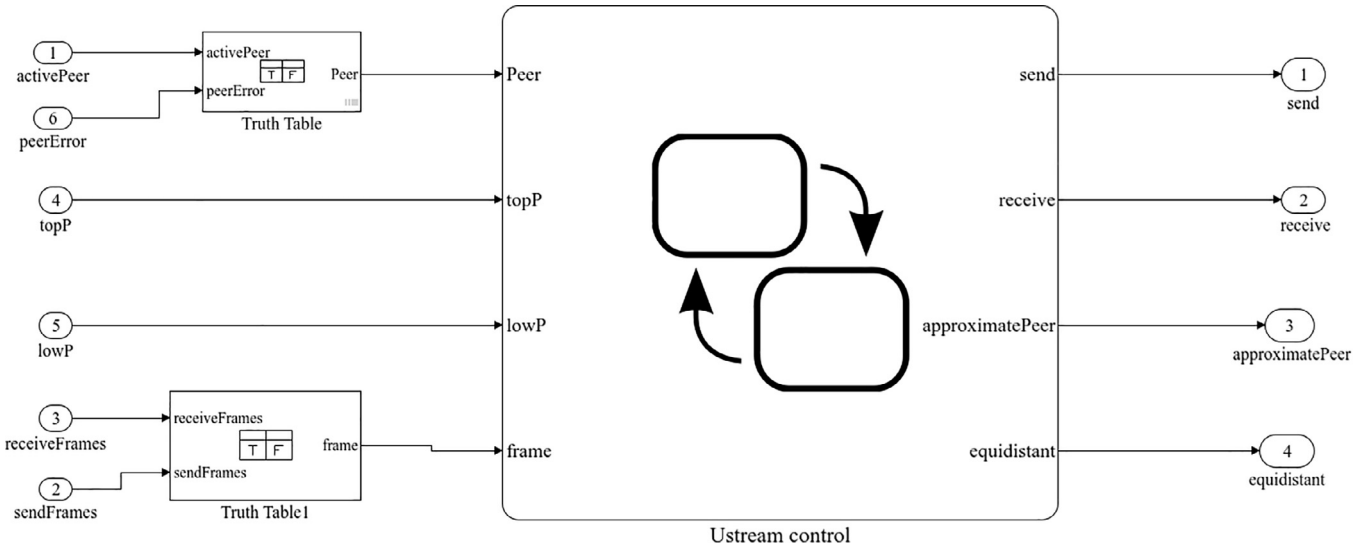

Fig. 9. Overlay-Network Module Control System.

Similarly, *receive* and *send* input variables from the parent module are processed by the control system using the truth table function and generate output variable *"frame"(send or receive)*. However, the input variables top and low peers are moved directly as input in the control system. The logical flow drives the parent module. It implements all steps involved in the overlay network module; it checks for the following: (i) whether the peers in the system are active or inactive, (ii) whether peers in the system can send or/and receive frames, (iii) the possibility of peers to become approximate peers and (iv) the possibility of peers within the same location to receive frames at the same time (that is, equidistant feature). Finally, the output results are displayed through the command windows.

6. Results and discussion

The outputs generated from the verification of UStream protocol are presented in this section. We start with the results of error-checking for all the modules. The simulation result shows that the video file was successfully transmitted from the encode module to the playout buffer module without error. The original video (see Fig. 10)a and the video output (See Fig. 10b) as observed are not perfect match due to loss of details. The occurrence of loss is evident in the traffic pattern depicted in Fig. 12. However, most of the salient features of the video are preserved. This error-free simulation result generated at the playout buffer module, implies a smooth transfer of video frames across all the modules in UStream protocol. Therefore, the error-free transmission validates the system and proves that the protocol is correct and consistent.

(a) Original Video

(b) Video Output

Fig. 10. UStream Protocol Output.

6.1. Verification results of schedule module

The results on classification of frames in schedule module validate the specification stated in Section 4.2 (case 1). Implementation of schedule module are driven by Boolean values: 0 represents false, 1 represents true and values between 0 and 1 represents the degree at which a specification is true. The result showed 100% validation for classification of frames in schedule module. This gives 100% assurance that all frames within schedule module will definitely be classified.

Also, the result for schedule module specification stated in Section 4.2 (case 2), suggests that prioritizing Iframe can only be true at the beginning ofa streaming session. Moreover, a gradual increase in movement between 0% and approximately 50% degree of validation occurred over ≈ a period of 1 s. Furthermore, Fig. 11 presents the verification for the specifications stated in Section 4.2 (cases 3–5). This figure shows that the time controller effectively transmitted frames (or delay frames) in the classified buffers to the traffic shaper. Although, some degree of oscillations between 0 and 1 over a period of 10 s were observed. This behavior points to transmission delay/failure at certain point within the system. The outcome of Section 4.2 (case 6) as depicted in Fig. 12, revealed 50% validation for constant flow of frames over 100 s with an interval break in transmission. This implies that the probability of achieving steady flow of video frames from the traffic shaper is 0.5 and the occurrence of interval may connote frame losses. Finally, the result of specification stated in Section 4.2 (case 7) validates the possibility of forwarding frames for a period of 20 s, despite the occurrence of errors at some point in the system.

6.2. Verification results of UStream module

The results for verification of Section 4.3 as showed in Fig. 13 revealed peers behavior in overlay module.The results on peers behaviour during sending or/and receiving frames as specified in Section 4.3 (cases 1–4) ascertains that the probability of peers

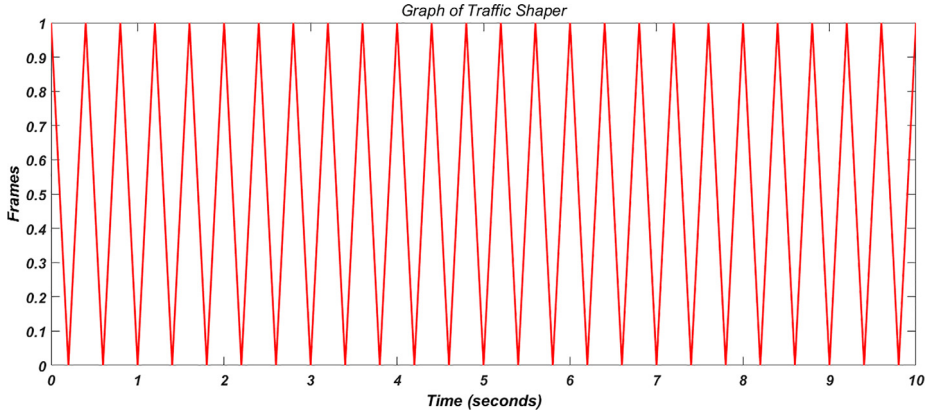

Fig. 11. The Graph of Traffic Shaper.

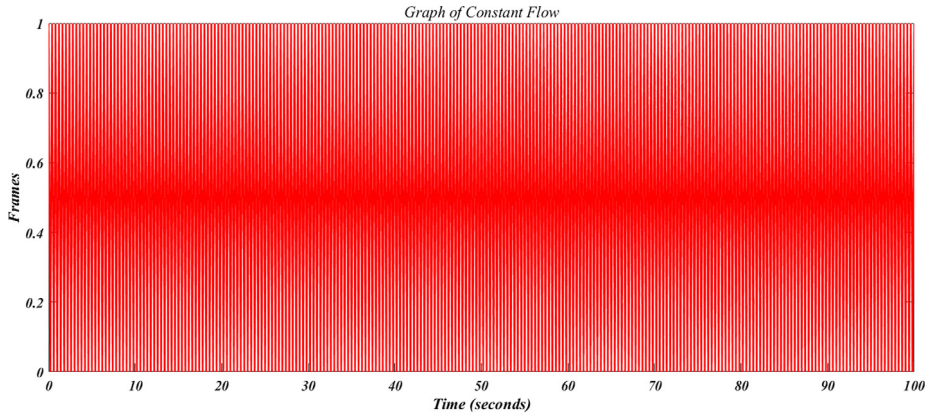

Fig. 12. The Graph of Constant Flow.

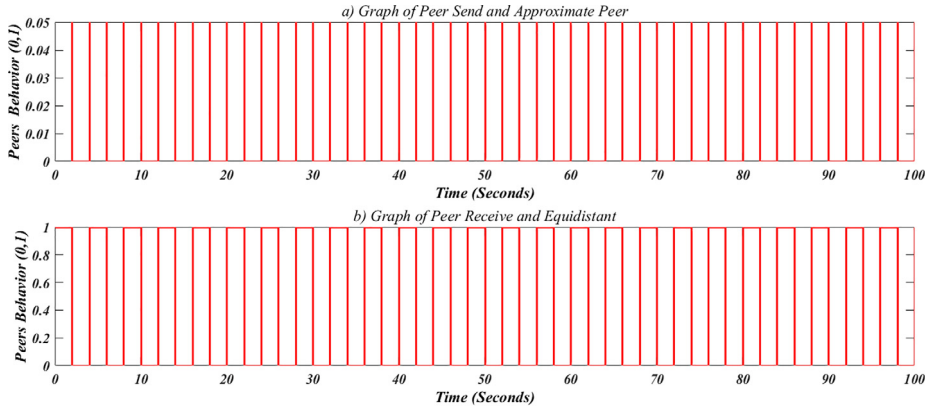

Fig. 13. The graph of Peers Behavior.

forwarding frames while receiving frames simultaneously is 50% (see Fig. 13a). On the other hand, the chances of peers receiving frames while sending frames is oscillatory between 0 and 1(as shown in Fig. 13b). However, a few consistent interval transmission failures, which may indicate delay in exchanging frames among peers, were observed. The result on approximate peers as specified in Section 4.3 (case 5) gives 50% validation with periodic failures(as displayed in Fig. 13a). This showed that the possibility of peers becoming an approximate is 50% and the periodic failures observed are as a result of heavy traffic load upon the approximate peer (since the original peers is down, the approximate caters for more peers). Lastly, result on equidistant between peers as specified in Section 4.3 (case 6) is oscillatory between true and false which proves that peers can be equidistant or otherwise(see Fig. 13b).

7. Conclusions

This paper presents the system analysis and verification of a P2P streaming protocol (UStream protocol) using model checking. The behavior of schedule and overlay network modules of the UStream protocol were specified using CTL. Each specification was verified using SDV within MATLAB simulation tool to check whether the temporal properties satisfy all the possible behaviors of the protocol. The simulation results proved that UStream protocol can achieve effective transmission of video files across peer-to-peer networks. In addition, the results revealed that the protocol can dynamically adapt to peer churn situation with 50% peers becoming approximate peers. Also, there is a 50% probability that peers are able to forward and receive frames simultaneously. Furthermore, the verification results revealed consistency in the scheme and presents a best case validation at 100% for some specifications and average case at 50% in other specifications. However, transmission delay/failure and frame losses were encountered in some case studies. Further research efforts will be targeted towards UStream protocol re-engineering for reduced transmission delay and frame loss as well as testing with the scalable video coding compressing technique.

Acknowledgments

This research is supported by 2014/2015 Tertiary Education Trust Fund (TeTFund) Academic Staff Training and Development (AST&D) Grant of Federal University of Agriculture, Abeokuta, Nigeria.

References

Akbari, A., Rosell, J., 2017. Physics-based motion planning with temporal logic specifications. IFAC-PapersOnLine 50 (1), 8993–8999.
Alessandria, E., Gallo, M., Leonardi, E., Mellia, M., Meo, M., 2011. Impact of adverse network conditions on P2P-tv systems: Experimental evidence. Comput. Netw. 55 (9), 2035–2050.
Apostolopoulos, J., Wee, S., 1999. Video Compression Standards, Wiley Encyclopedia of Electrical and Electronics Engineering. John Wiley and Sons Inc, New York.
Armando, A., Carbone, R., Compagna, L., Li, K., Pellegrino, G., 2010. Model-checking driven security testing of web-based applications. In: Third International Conference on Software Testing, Verification, and Validation Workshops (ICSTW), IEEE, pp. 361–370.
Armando, A., Ponta, S.E., 2009. Model checking of security-sensitive business processes. In: International Workshop on Formal Aspects in Security and Trust. Springer, pp. 66–80.

Ayatollahi, H., Khansari, M., Rabiee, H., 2018. A push-pull network coding protocol for live peer-to-peer streaming. Comput. Netw. 130, 145–155.
Barekatain, B., Khezrimotlagh, D., Maarof, M., Ghaeini, H., Quintana, A.A., Cabrera, A., 2015. Efficient P2P live video streaming over hybrid wmns using random network coding. Wireless Pers. Commun. 80 (4), 1761–1789.
Barekatain, B., Khezrimotlagh, D., Maarof, M., Ghaeini, H., Salleh, S., Quintana, A.A., Akbari, B., Cabrera, A.T., 2013. Matin: a random network coding based framework for high quality peer-to-peer live video streaming. PloS One 8 (8), e69844.
Basin, D., Caleiro, C., Ramos, J., Viganò, L., 2011. Distributed temporal logic for the analysis of security protocol models. Theoret. Comput. Sci. 412 (31), 4007–4043.
Batt, G., Belta, C., Weiss, R., 2008. Temporal logic analysis of gene networks under parameter uncertainty. IEEE Trans. Autom. Control 53 (Special Issue), 215–229.
Bellini, P., Mattolini, R., Nesi, P., 2000. Temporal logics for real-time system specification. ACM Comput. Surveys (CSUR) 32 (1), 12–42.
Biere, A., Cimatti, A., Clarke, E., Zhu, Y., 1999. Symbolic model checking without bdds. In: International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Springer, pp. 193–207.
Brienza, S., Cebeci, S., Masoumzadeh, S., Hlavacs, H., Özkasap, Ö., Anastasi, G., 2016. A survey on energy efficiency in P2P systems: file distribution, content streaming, and epidemics. ACM Comput. Surv. 48 (3), 36.
Chen, Y., Zhang, B., Chen, C., Chiu, D., 2014. Performance modeling and evaluation of peer-to-peer live streaming systems under flash crowds. IEEE/ACM Trans. Network. (TON) 22 (4), 1106–1120.
Clarke, E., Grumberg, O., Jha, S., Lu, Y., Veith, H., 2003. Counterexample-guided abstraction refinement for symbolic model checking. J. ACM (JACM) 50 (5), 752–794.
Clarke, E., Grumberg, O., Peled, D., 1999. Model Checking. MIT Press.
Datta, A., Derek, A., Mitchell, J., Roy, A., 2007. Protocol composition logic (PCL). Electron. Notes Theor. Comput. Sci. 172, 311–358.
Davies, R., 2017. A temporal logic approach to binding-time analysis. J. ACM (JACM) 64 (1), 1:1–1:45.
Dixon, C., Gago, M.-C.F., Fisher, M., van der Hoek, W., 2007. Temporal logics of knowledge and their applications in security. Electron. Notes Theor. Comput. Sci. 186, 27–42.
Dixon, C., Winfield, A.F., Fisher, M., Zeng, C., 2012. Towards temporal verification of swarm robotic systems. Rob. Auton. Syst. 60 (11), 1429–1441.
Eittenberger, P.M., Herbst, M., Krieger, U.R., 2012. Rapidstream: P2p streaming on android. In: Packet Video Workshop (PV), 2012 19th International, IEEE, pp. 125–130.
Fallica, B., Lu, Y., Kuipers, F., Kooij, R., Mieghem, P.V., 2008. On the quality of experience of sopcast. In: The Second International Conference on Next Generation Mobile Applications, Services and Technologies (NGMAST). IEEE, pp. 501–506.
Fiandrotti, A., Sheikh, A., Magli, E., 2012. Towards a P2P videoconferencing system based on low-delay network coding. In: Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European. IEEE, pp. 1529–1533.
Ghaeini, H.R., Akbari, B., 2014. Peer-to-peer adaptive forward error correction in live video streaming over wireless mesh network. In: International Conference on Wired/Wireless Internet Communications. Springer, pp. 109–121.
Ghaeini, H.R., Akbari, B., Barekatain, B., 2013. An adaptive packet loss recovery method for peer-to-peer video streaming over wireless mesh network. In: Emerging Technologies for Information Systems, Computing and Management. Springer, pp. 713–721.
Ghaeini, H.R., Akbari, B., Barekatain, B., Trivino-Cabrera, A., 2016. Adaptive video protection in large scale peer-to-peer video streaming over mobile wireless mesh networks. Int. J. Commun Syst 29 (18), 2580–2603.
Golston, J., 2004. Comparing media codecs for video content. In: Embedded Systems Conference, San Francisco.
Gomes, P., Campos, S., Vieira, A., 2012. Verification of P2P live streaming systems using symmetry-based semiautomatic abstractions. In: International Conference on High Performance Computing and Simulation. pp. 343–349. https://doi.org/10.1109/HPCSim.2012.6266935.
Gonçalves, G.D., Cunha, Í., Vieira, A., Almeida, J., 2016. Predicting the level of cooperation in a peer-to-peer live streaming application. Multimedia Syst. 22 (2), 161–180.
Hammami, C., Jemili, I., Gazdar, A., Belghith, A., Mosbah, M., 2014. Hybrid live P2P streaming protocol. Procedia Comput. Sci. 32, 158–165.
Haskell, B., Puri, A., 2012. Mpeg video compression basics. In: The MPEG Representation of Digital Media. Springer, pp. 7–38.
Hei, X., Liang, C., Liang, J., Liu, Y., Ross, K., 2007. A measurement study of a large-scale P2P IPTV system. IEEE Trans. Multimedia 9 (8), 1672–1687.
Hu, H., Guo, Y., Liu, Y., 2011. Peer-to-peer streaming of layered video: efficiency, fairness and incentive. IEEE Trans. Circuits Syst. Video Technol. 21 (8), 1013–1026.
Jiang, M., Ma, Z.-F., Niu, X.-X., Yang, Y.-X., 2011. Video watermarking scheme based on MPEG-2 for copyright protection. Procedia Environ. Sci. 10, 843–848.
Le, L., Nguyen, S., Nguyen, N.X., Dang, T., 2017. A principle of adaptively grouping frames on lossless medical video compression using ideal cross-point regions. In: International Conference on the Development of Biomedical Engineering in Vietnam. Springer, pp. 19–23.
Liao, X., Jin, H., Liu, Y., Ni, L., 2007. Scalable live streaming service based on interoverlay optimization. IEEE Trans. Parallel Distrib. Syst. 18 (12), 1663–1674.
Liao, X., Jin, H., Liu, Y., Ni, L., Deng, D., 2006. Anysee: Peer-to-peer live streaming. In: 25th International Conference on Computer Communications. IEEE, pp. 1–10.
Liu, J., Ozay, N., Topcu, U., Murray, R., 2013. Synthesis of reactive switching protocols from temporal logic specifications. IEEE Trans. Autom. Control 58 (7), 1771–1785.
Magharei, N., Rejaie, R., 2009. Prime: peer-to-peer receiver-driven mesh-based streaming. IEEE/ACM Trans. Network. (TON) 17 (4), 1052–1065.
Merani, M., Natali, L., 2016. Adaptive streaming in P2P live video systems: a distributed rate control approach. ACM Trans. Multimedia Comput. Commun. Appl. 12 (3), 46.
Miller, A., Donaldson, A., Calder, M., 2006. Symmetry in temporal logic model checking. ACM Comput. Surveys (CSUR) 38 (3), 8.
Naiem, A., El-Beltagy, M., 2016. Nat constraints management in tree-based P2P live streaming systems. In: Proceedings of the 10th International Conference on Informatics and Systems. ACM, pp. 292–297.
Ohm, J.-R., Sullivan, G., Schwarz, H., Tan, T., Wiegand, T., 2012. Comparison of the coding efficiency of video coding standards–including high efficiency video coding (HEVC). IEEE Trans. Circuits Syst. Video Technol. 22 (12), 1669–1684.
Ojo, O., Oluwatope, A., Ajadi, S., 2017. Dynamical analysis of an internet-based video system. IFAC-PapersOnLine 50 (2), 221–226.
Ojo, O., Oluwatope, A., Ogunsola, O., 2015. Ustream: Ultra-metric spanning overlay topology for peer-to-peer streaming systems. In: 2015 IEEE International Symposium on Multimedia (IEEEISM-2015). IEEE, Miami, Florida, USA, pp. 601–604.
Purandare, D., Guha, R., 2007. An alliance based peering scheme for P2P live media streaming. IEEE Trans. Multimedia 9 (8), 1633–1644.
Richardson, I., 2010. The H.264 Advanced Video Compression Standard. Wiley. Ch. 6, pp. 169–171.
Rozier, K.Y., 2011. Linear temporal logic symbolic model checking. Comput. Sci. Rev. 5 (2), 163–203.
Sandvik, P., Sere, K., 2011. Formal analysis and verification of peer-to-peer node behaviour. In: The Third International Conference on Advances in P2P System. Citeseer, pp. 47–57.
Seeling, P., Reisslein, M., 2014. Video traffic characteristics of modern encoding standards: H. 264/AVC with SVC and MVC extensions and h. 265/HEVC. Sci. World J. 2014, 1–16.
Shen, H., Lin, Y., Li, J., 2015. A social-network-aided efficient peer-to-peer live streaming system. IEEE/ACM Trans. Network. 23 (3), 987–1000.
SopCast, 2007. Sopcast – free P2P internet tv. Date accessed: 15/06/2018. URLhttp://www.sopcast.org.
Sullivan, G., 2005. Overview of international video coding standards (preceding h. 264/avc). In: ITU-T VICA workshop, Geneva.
Ullah, I., Doyen, G., Bonnet, G., Gaïti, D., 2012a. An autonomous topology management framework for qos enabled P2P video streaming systems. In: Network and service management (CNSM), 2012 8th international conference and 2012 workshop on systems virtualiztion management (SVM). IEEE, pp. 126–134.
Ullah, I., Doyen, G., Bonnet, G., Gaïti, D., 2012b. A survey and synthesis of user behavior measurements in P2P streaming systems. IEEE Commun. Surveys Tutorials 14 (3), 734–749.
Ullah, I., Doyen, G., Bonnet, G., Gaïti, D., 2013. Towards user-aware peer-to-peer live video streaming systems. In: Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on. IEEE, pp. 920–926.
Wallach, D.S., 2003. A survey of peer-to-peer security issues. In: Software Security, Theories and Systems. Springer, pp. 42–57.
Wang, M., Li, B., 2007. R2: random push with random network coding in live peer-to-peer streaming. IEEE J. Sel. Areas Commun. 25 (9).
Wang, W., Farid, H., 2006. Exposing digital forgeries in video by detecting double mpeg compression. In: Proceedings of the 8th workshop on Multimedia and security. ACM, pp. 37–47.
Xiao, Z., Ye, F., 2008. New insights on internet streaming and iptv. In: Proceedings of the 2008 international conference on Content-based image and video retrieval. ACM, pp. 645–654.
Yang, C., Wu, C., Yang, Z., Liu, T., Yin, Z., Liu, Y., Mao, X., 2016. Enhancing industrial video surveillance over wireless mesh networks. In: 25th International Conference on Computer Communication and Networks. IEEE, pp. 1–9.
Yao, H., Song, S., Qin, C., Tang, Z., Liu, X., 2017. Detection of double-compressed h. 264/AVC video incorporating the features of the string of data bits and skip macroblocks. Symmetry 9 (12), 313.
Yao, X.-W., Wang, W.-L., Yang, S.-H., Cen, Y.-F., Yao, X.-M., Pan, T.-Q., 2014. IPB-frame adaptive mapping mechanism for video transmission over IEEE 802.11e WLANs. ACM SIGCOMM Comput. Commun. Rev. 44 (2), 5–12.
Zatt, B., Porto, M., Scharcanski, J., Bampi, S., 2010. Gop structure adaptive to the video content for efficient h. 264/avc encoding. In: 2010 17th IEEE International Conference on Image Processing. IEEE, pp. 3053–3056.
Zhang, X., Liu, J., Li, B., Yum, T., 2005. Coolstreaming/donet: a data-driven overlay network for peer-to-peer live media streaming. In: 24th Annual Joint Conference of the IEEE Computer and Communications Societies. vol. 3. IEEE, pp. 2102–2111.