2021 European Control Conference (ECC)
June 29 - July 2, 2021. Rotterdam, Netherlands

Synchronizing Live Video Streaming Players Via Consensus

Gioacchino Manfredi, Luca De Cicco, Saverio Mascolo

Abstract— **Video streaming is the primary source of the global Internet traffic. Social media applications allow users to experience live streaming events together, even though not being in the same physical place. The current online video delivery architecture cannot ensure a synchronized video playback among geographically distributed users. Unsynchronized playback can become evident and negatively impact the users' feelings of togetherness. In this paper, we show that the well-known consensus problem of simple integrators with saturated inputs is an appropriate mathematical framework to design a distributed playback synchronization mechanism. Furthermore, we propose a leader-following approach to ensure synchronization among users. Simulations on different network topologies confirm that the proposed approach is effective at enforcing asymptotic synchronization without having detrimental effects on the users' perceived quality.**

I. Introduction and background

Today, users are increasingly adopting video streaming services instead of classical TV broadcast channels to consume a broad spectrum of media going from on-demand content (e.g., movies, TV series) to live events (e.g., sport, news).

However, media delivery over the Internet is affected by two main issues that impair the users' perceived quality: 1) video playback might stall or video quality might degrade impacting *service smoothness* [1]; 2) in live events, geographically distributed users might watch the same content but with different playback times having detrimental effects on the *service togetherness*, i.e., the level of satisfaction that users perceive when feeling that the service is experienced together with a number of users. When the streams are not synchronised, some clients may watch a certain event (e.g. a goal in a football match) before the others, thus negatively impacting users' feelings of togetherness.

In order to overcome the first issue, the video content bitrate must be made adaptive. To this end, the standard currently employed to stream media content, i.e., the MPEG-Dynamic Adaptive Streaming over HTTP (MPEG-DASH or DASH) [2] requires that each video is compressed as follows. The video is firstly divided into *segments* or chunks of fixed duration τ (typically 1 to 10 seconds) and then each segment is compressed to originate a number of *levels*, characterized by different encoding bitrates and video resolutions. This way, the clients can dynamically choose the suitable level to fetch according to a controller denoted as the *Adaptive BitRate (ABR)* algorithm. Its goal is to maximize the overall perceived quality given the Internet available bandwidth while avoiding rebuffering events. The design of ABR control algorithms has been extensively studied in the literature with contributions also set in the control community [3], [4]. On the other hand, the issue of unsynchronized video streams affecting service togetherness is still unsolved.

Gioacchino Manfredi, Luca De Cicco, and Saverio Mascolo are with the Dipartimento di Ingegneria Elettrica e dell'Informazione at Politecnico di Bari, Via Orabona 4, 70125, Bari, Italy Emails: gioacchino.manfredi@poliba.it, luca.decicco@poliba.it, mascolo@poliba.it

To the best of our knowledge, this research issue has never been addressed by the control system community. In fact, the published literature is all set in the research fields of multimedia and computer networks and offers few works mostly using centralized approaches [5], [6]. The drawback of such solutions is that they do not scale with the number of users. It is worth mentioning that the only works proposing a decentralized approach resort to heuristics that do not allow a rigorous analysis of system properties [7], [8].

In this paper, we propose a fully distributed approach for video playout synchronization in which each client in the network can exchange its own asynchrony value only with that of some neighbouring clients. This work makes two key contributions. First, it provides a mathematical model of the playback time of an event that clearly explains the causes for synchronization issues (Section II). Second, it shows that the problem of synchronizing video players can be expressed as a consensus problem involving integrators with saturated control inputs (Section III). As a matter of fact, the application of consensus theory to the field of video streaming is not new in the literature, e.g., in [9] consensus is leveraged to provide video quality fairness.

In our work, the synchronization algorithm is decoupled from the ABR algorithm, which is left unchanged. The control input leveraged to synchronize clients is the playback rate, an approach denoted as *Adaptive Media Playout (AMP)*. In practice, the playback rate, i.e. the speed of reproduction of the video, can be slightly adjusted to control the playback time. Several studies on the user's perceived quality, or *Quality of Experience (QoE)*, have shown that the playback rate can be increased/decreased by only a small amount to prevent the QoE from being negatively impacted [10], [11]. Thus, our proposed approach, with the playback rate bounded in a given interval, allows asymptotic convergence of the clients to the same playback time (Section IV).

II. Playback time model

This section is organized as follows. First, a model of the playback time is provided in Section II-A. Based on this model, in Section II-B we show the causes of synchronization issues among users. Finally, in Section II-C we present a model of the AMP approach that can be leveraged to synchronize players.

Authorized licensed use limited to: Politecnico di Bari. Downloaded on December 13,2023 at 10:29:52 UTC from IEEE Xplore. Restrictions apply.

A. *Video playback time model*

This section presents a model of the playback time of a user watching a live event that starts at time $t = 0$. Notice that the live event time coincides with the real time t, thus we will use the term *time* end *event time* interchangeably.

DASH-compliant live streaming systems (e.g., YouTube, Facebook, etc.) produce live videos. A camera captures a live scene that is compressed in real-time by an encoder. Each τ seconds this element produces video segments with the same video content encoded at different bitrate levels l_i belonging to a discrete set $\mathscr{L} = \{l_1, \dots, l_M\}$ ($l_i < l_{i+1}$). A *circular buffer* accommodates the k segments most recently produced by the encoder for each of the levels $l_i \in \mathscr{L}$. Notice that the client can only access the segments stored in the circular buffer. Thus, at any given time t, a user can access segments containing video content representing the scene captured in a time window $W(t)$ defined as follows. We denote with $s(t)$ the segment index that contains the event time t:

$$s(t) = \left\lfloor \frac{t}{\tau} \right\rfloor \tag{1}$$

where $\lfloor \cdot \rfloor$ is the *floor* operator that maps a real number x to the greatest natural number less than x. Notice that the segment $s(t)$ contains the event scene belonging to the time interval $[s(t)\tau, (s(t) + 1)\tau[$. Equipped with this notation, it is easy to see that the window $W(t)$ representing the event time currently stored in the circular buffer is:

$$W(t) = [s(t)\tau - k\tau, s(t)\tau[\tag{2}$$

When a user joins a live event at time t_J, it immediately starts downloading the oldest segment available in the circular buffer, i.e. the one with index $s(t_J) - k$. The retrieved segments are temporarily stored in the client's playout buffer whose level, measured in seconds, is denoted with $q(t)$. The ABR control algorithm decides the video level $l(t) \in \mathscr{L}$ to be downloaded for each segment. Such algorithms typically wait for the playout buffer level $q(t)$ to reach a certain value q_L that is considered enough to mitigate the rebuffering events occurring when the buffer gets completely depleted ($q(t) = 0$). The playout buffer level $q(t)$ can be modeled as an integrator [1]:

$$\dot{q}(t) = f(t) - p(t) \tag{3}$$

where $f(t)$ is the fill rate, i.e. the rate of seconds of video received and stored in the playout buffer and $p(t)$ is the *playback rate*, i.e. the rate of seconds of video drained by the playout buffer and fed to the decoder. In [1] we show that $f(t) = r(t)/l(t)$, where $r(t)$ is the download rate that depends on the time-varying network bandwidth measured in bytes/s.

Under normal operation, when the video is playing the playback rate $p(t)$ is equal to 1. This indicates that the video playback is not slowed down nor sped up. Of course, when the player is in pause the playback rate is 0. Thus, under normal conditions it holds:

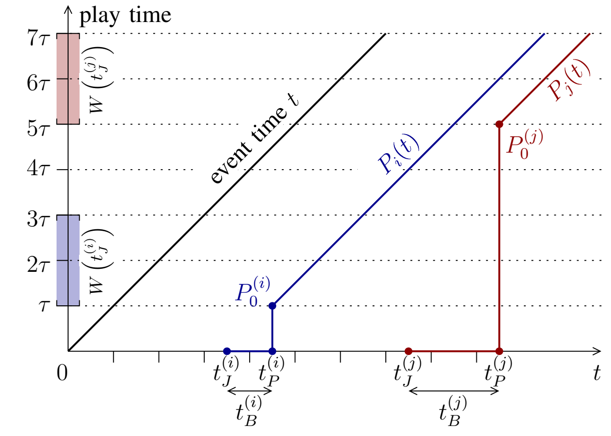

Fig. 1. Playback time dynamics of two users i and j joining at different time instants

$$p(t) = \begin{cases} 1 & \text{playing} \\ 0 & \text{paused} \end{cases} \tag{4}$$

We are now ready to derive the model of the playback time $P(t)$. When the user joins the live event at time $t = t_J$, it immediately starts downloading the segment with playback time:

$$P_0 = s(t_J)\tau - k\tau \tag{5}$$

which corresponds to the first segment stored in the producer's circular buffer at time t_J. Recall that the player is paused ($p(t) = 0$) until the queue level $q(t)$ surpasses q_L. The time t_B needed to fill the playout buffer to reach q_L can be found by integrating (3) and imposing the constraint $q(t_B) = q_L$. Thus, t_B depends both on the download rate $r(t)$, which can be considered as a disturbance, and on $l(t)$ that is the output of the ABR control algorithm.

The player starts reproducing the video at time $t_P = t_J + t_B$. By definition, for $t > t_P$, the playback time dynamics is given by:

$$\dot{P}(t) = p(t) \tag{6}$$

with initial condition $P(t_P) = P_0$, which leads to:

$$P(t) = P_0 + t - t_P \tag{7}$$

B. *The synchronization issue*

To shed more light on the causes of synchronization issues, refer to Figure 1 that depicts the play time of two users, denoted with i and j, that are interested in watching the same live streaming event. The figure shows the case of a live video streaming system using a circular buffer containing the two most recently produced segments, i.e., $k = 2$. The user i is the first to join the event at time $t_J^{(i)}$, after which the ABR algorithm immediately starts to download and store in the playout buffer the video chunks contained in the circular buffer identified by the time window $W(t_J^{(i)})$ (blue shaded area in the figure). When the buffer length reaches q_L after $t_B^{(i)}$ seconds, the playback on the user's device starts, i.e.

Authorized licensed use limited to: Politecnico di Bari. Downloaded on December 13,2023 at 10:29:52 UTC from IEEE Xplore. Restrictions apply.

$p(t) = 1$, $t \geq t_P^{(i)}$ where $t_P^{(i)} = t_J^{(i)} + t_B^{(i)}$. Notice that the playback starts with a play time $P_0^{(i)}$, as defined in (5), which represents the play time of the video at the beginning of the first chunk stored in the buffer, i.e. the beginning of the window $W(t_J^{(i)})$. The user j joins the same live event at time $t_J^{(j)}$ after the user i. In the same way, the ABR algorithm fetches the video chunks identified by the time window $W(t_J^{(j)})$ (red shaded area in the figure) and fills the playout buffer until it reaches the minimum level q_L, which is needed to begin the playback on the user's device[1]. However, the time $t_B^{(j)}$ needed to complete this operation is different in general from $t_B^{(i)}$. Therefore, the playback at user j will start at time $t_P^{(j)} = t_J^{(j)} + t_B^{(j)}$ from a play time $P_0^{(j)}$. Finally, the playback time of user i can be denoted as $P_i(t) = P_0^{(i)} + t - t_P^{(i)}$ while the playback time of user j is $P_j(t) = P_0^{(j)} + t - t_P^{(i)}$. Hence, the asynchrony of the two users is

$$P_i(t) - P_j(t) = [(P_0^{(i)} - t_J^{(i)}) - (P_0^{(j)} - t_J^{(j)})] + (t_B^{(j)} - t_B^{(i)}) \quad (8)$$

where $P_0^{(i)} - t_J^{(i)}$ represents the distance of $t_J^{(i)}$ from the play time $P_0^{(i)}$ of the video at the beginning of the first chunk stored in the buffer. The same explanation can be given to $P_0^{(j)} - t_J^{(j)}$ for user j. As a consequence, if these two distances are equal as well as the buffering times, then $P_i(t) - P_j(t) = 0$ and the two users will be synchronized. Therefore, it is immediate to deduce that the asynchrony between two users depends on their join and buffering times.

C. The Adaptive Media Playout model

In this section, we present the AMP approach, which consists of allowing the playback rate to be slightly varied around the nominal value 1, i.e. the playback could be slightly slowed down or sped up of a term $u(t)$. Notice that $u(t)$ must be small enough to be barely noticeable by the user so that the user's QoE is not affected [10], [11]. To this end, we redefine the playback rate as follows:

$$p_r(t) = p(t)(1 + u(t)) \quad (9)$$

where $u(t) \in [-\delta, \delta]$, with δ small enough, and $p(t)$ is the playback rate (4). Thus, the playback time now becomes:

$$P(t) = P_0 + \int_{t_P}^{t} p(\xi)(1 + u(\xi))d\xi \quad \forall\, t \geq t_P \quad (10)$$

Suppose N clients are watching the same live streaming event and assume none of them is experiencing a rebuffering event[2] (i.e., $p(t) = 1\ \forall\, t \geq t_P$). Then, for each client i and $\forall\, t \geq t_P^{(i)}$, the playing time is

$$P_i(t) = P_0^{(i)} + \int_{t_P^{(i)}}^{t} (1 + u_i(\xi))d\xi \quad (11)$$

[1]Notice that, in general, the value of q_L might be different for the two users.

[2]Notice that this is a well-posed and realistic assumption since, as already mentioned, ABR algorithms are specifically designed to avoid such events.

The playback rate variations $u_i(t)$, $i = 1, ..., N$, are the control variables on which we will act to achieve synchronization among users. This goal is attained when $P_i(t) \to P_j(t)$ $\forall\, i, j = 1, ..., N$ asymptotically.

III. The Proposed Synchronization Approach

Before presenting our solution, let us start by describing its design requirements: (R1) the system must be horizontally scalable, i.e., it could be used for large events; (R2) it has to be implementable with technologies already available and used by the media distribution industry. To meet the design criteria (R1) a decentralized control approach must be used. The requirement (R2) can be met as follows: to implement synchronization messages to be exchanged directly among users, without the need of a central server, the WebRTC open standard can be used.

Figure 2 depicts the architecture of the proposed approach for playback time synchronization. In particular, the ABR algorithm running at the i-th client, on the basis of some information such as, e.g., the estimated available bandwidth and the playout buffer level $q_i(t)$, dynamically selects the level $l_i(t)$ and requests it to the video producer server. Therefore, the server delivers the video segments encoded at the level selected by the ABR component. Once these segments reach the client, they are stored in the playout buffer waiting to be played. The *synchronization controller*, the focus of this work, decides the most suitable value of the playback rate $p_{r,i}(t)$ at which the video has to be played at the client's screen. Finally, the decoder decompresses the video frames drained form the playout buffer and renders them on the client's screen at the playback rate $p_{r,i}(t)$ imposed by the synchronization controller. Before describing the proposed approach, let us introduce the following non restrictive assumption:

Assumption 1. Once each user $i = 1, \ldots, N$ starts the playback of the live streaming content, no rebuffering event occurs, which implies (i) $p_i(t) \neq 0\ \forall\, t \geq t_P^{(i)}$ and (ii) the ABR algorithm is able to avoid playout buffer depletion, i.e. $q_i(t) \geq 0, \forall\, t \geq t_P^{(i)}$.

The starting point of the proposed approach is to derive the dynamic model of the playing time evolution from (11). Let us define the initial delay of the i-th user as $T_{0i} = P_0^{(i)} - t_P^{(i)} < 0$. The synchronization algorithm starts at time $t = t_s$ when the N users attending the event have started the video playback, i.e. $p_i(t) = 1\, \forall\, t > t_s, \forall i \in \{1, \ldots, N\}$. Notice that for $t < t_s$ the synchronization algorithm is not active, thus it turns out that $u_i(t) = 0, \forall\, i \in \{1, \ldots, N\}$. As a consequence, the playback time can be obtained as follows:

$$P_i(t) = T_{0i} + t + \int_{t_P^{(i)}}^{t} u_i(\xi)d\xi = T_{0i} + t + \int_{t_s}^{t} u_i(\xi)d\xi \quad (12)$$

$\forall\, t \geq t_s$, where the last equality comes from the fact that $u_i(t) = 0$ for $t_P^{(i)} \leq t \leq t_s$. Hence, the dynamical model is

$$\dot{P}_i(t) = 1 + u_i(t) \quad (13)$$

Authorized licensed use limited to: Politecnico di Bari. Downloaded on December 13,2023 at 10:29:52 UTC from IEEE Xplore. Restrictions apply.

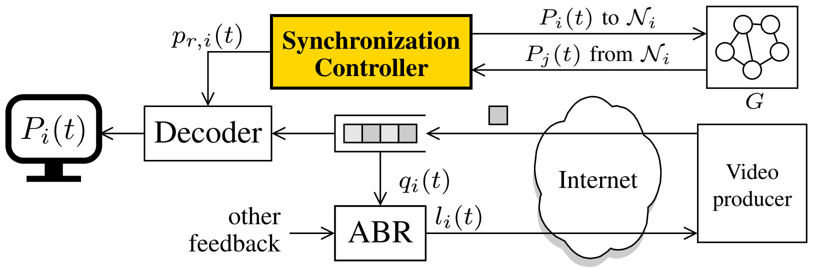

Fig. 2. The proposed synchronization control architecture

From the previous equation it is clear that the control variable on which the controller can act to address the problem is $u_i(t)$, $\forall t \geq t_s$. In order to obtain a set of integrators, we need to perform a change of coordinates: let us define $x_i(t) = P_i(t) - t$, thus $\dot{x}_i(t) = \dot{P}_i(t) - 1 = u_i(t)$, $\forall t \geq t_s$. As a consequence, controlling the models of all the clients is equivalent to controlling a set of integrators as defined in the following:

$$\begin{cases} \dot{P}_1(t) = 1 + u_1(t) \\ \vdots \\ \dot{P}_N(t) = 1 + u_N(t) \end{cases} \equiv \begin{cases} \dot{x}_1(t) = u_1(t) \\ \vdots \\ \dot{x}_N(t) = u_N(t) \end{cases} \tag{14}$$

Notice that $x_i(t)$ is the temporal delay of user i with respect to the current playout time transmitted by the provider.

In order to model the proposed approach as a consensus problem, let us define a directed graph (or digraph) $G(\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ is the set of nodes, also identified simply by its indices $i = 1, \ldots, N$, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges. An edge (v_i, v_j) denotes the information flow from node i to node j. Moreover, the neighbours of node v_i are contained in the set $\mathcal{N}_i = \{v_j \in \mathcal{V} : (v_j, v_i) \in \mathcal{E}\}$. In this setting, it is possible to model the clients as the nodes $i \in \mathcal{V}$ of the graph G, where each node i is associated to the corresponding state $x_i(t)$ and can receive information from a given number of neighbours $\mathcal{N}_i$. If we assume that the control inputs $u_i(t)$ are unbounded, then for a set of integrators we can use the well-known control strategy [12]:

$$u_i(t) = \sum_{j \in \mathcal{N}_i} a_{ij}(x_j - x_i) = \sum_{j \in \mathcal{N}_i} a_{ij}(P_j - P_i) \tag{15}$$

where $A = A(G) = (a_{ij})$ is the adjacency matrix given by $a_{ij} = 1$ if $(j, i) \in \mathcal{E}$ and $a_{ij} = 0$ otherwise.

The resulting system is as follows

$$\dot{\boldsymbol{x}}(t) = -L\boldsymbol{x}(t), \ \ \forall t \geq t_s \tag{16}$$

where $\boldsymbol{x}(t) = [x_1(t), ..., x_N(t)]^\top$ is the stack vector of all the agents' states and L is the Laplacian matrix associated to A. If G is strongly connected, $-L$ has eigenvalues such that $-\lambda_{N-1} \leq -\lambda_{N-2} \leq \cdots < -\lambda_0 \leq 0$, where $\lambda_0 = 0$ and $\text{rank}(L) = N - 1$. A well-known result of [12] is that the control (15) globally asymptotically solves the consensus problem for the set of integrators in (14). In other words, it results that (16) is stable and $\exists\, \alpha \in \mathbb{R}$ s.t. it converges to the equilibrium point $\bar{\boldsymbol{x}} = \alpha\mathbf{1}$, i.e., $\bar{x}_i = \alpha, \forall i \in \mathcal{V}$. If G is balanced, i.e., the in-degree is equal to the out-degree for each node, then $\alpha = \mathbb{E}[x_i(t_s)] = \frac{1}{N}\sum_{i=1}^N x_i(t_s)$. Thus, when the consensus is reached, $x_i(t) = P_i(t) - t = T_{0i} + \int_{t_s}^t u_i(\xi)d\xi = \alpha$ and thus $P_i(t) = t + \alpha$ for all i. In other words, if we can solve the consensus problem, we also solve the playback time synchronization problem. Notice that when G is balanced $\alpha = \mathbb{E}[x_i(t_s)]$, so it turns out that $P_i(t)$ converges to $t + \mathbb{E}[T_{0i}]$, where the second term represents the average of the initial delays of the clients.

However, the previous approach entailed the unrealistic assumption of unbounded control inputs $u_i(t)$. To tackle this issue, we define the following saturation function:

$$\sigma(x) = \begin{cases} \delta & x > \delta \\ x & -\delta \leq x \leq \delta \\ -\delta & x < -\delta \end{cases} \tag{17}$$

that, in the case of an N-element vector, will define $\sigma(\boldsymbol{x}) = [\sigma(x_1), \sigma(x_2), \ldots, \sigma(x_N)]^\top$.

Suppose that $u_i(t) \in [-\delta, \delta]$, $\forall i \in \mathcal{V}$, then

$$\dot{\boldsymbol{x}} = \sigma(\boldsymbol{u}) \tag{18}$$

At this point, using the same control strategy as in (15), $\dot{x}_i(t) = \sigma(u_i(t)) = \sigma\left(k_i \sum_{j \in \mathcal{N}_i} a_{ij}(x_j - x_i)\right)$, where $k_i > 0$, $\forall i \in \mathcal{V}$ are controller gains. Therefore it results that

$$\dot{\boldsymbol{x}} = \sigma(-KL\boldsymbol{x}) \tag{19}$$

where $K = \text{diag}\{k_1, \ldots, k_N\}$.

Before introducing the main result, we recall the following lemma from [13]:

Lemma 1. *Given a strongly connected digraph G, the associated Laplacian matrix L has a simple eigenvalue in zero and all the nonzero eigenvalues have positive real part. Let $\boldsymbol{r} = [r_1, \ldots, r_N]^\top > 0$ be a left eigenvector of L associated to the zero eigenvalue, i.e., $\boldsymbol{r}^\top L = L^\top \boldsymbol{r} = 0$, and let $R = \text{diag}\{r_1, \ldots, r_N\}$, $Q = RL + L^\top R$. Then $R > 0$, $Q \geq 0$ and the kernel of Q has dimension 1 and is given by $\text{span}\{\mathbf{1}_N\}$.*

We are now ready to introduce the following:

Theorem 2. *Consider a multi-agent system represented by a graph G whose dynamics are described by (18). Suppose*

Authorized licensed use limited to: Politecnico di Bari. Downloaded on December 13,2023 at 10:29:52 UTC from IEEE Xplore. Restrictions apply.

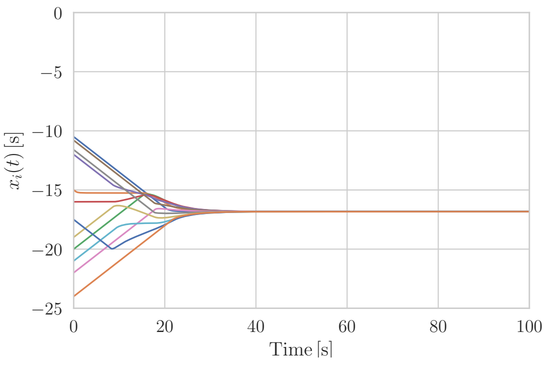

Fig. 3. States dynamics $x_i(t)$ for the ring topology

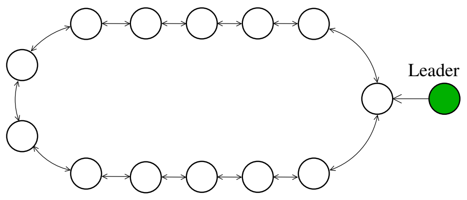

Fig. 4. Ring topology with a leader node

each agent applies the control (15) and assume that G is a strongly connected and directed graph. Then the control (15) globally asymptotically solves a consensus problem.

From Theorem 2, we can state the following:

Corollary 3. *Consider N users watching the same live streaming event. Suppose the users can receive the playback times $P_j(t)$ from a set of clients $j \in \mathcal{N}_i$ according to an established strongly connected digraph whose adjacency matrix is $A = (a_{ij})$. Then, if the playback rate is set as $p_r^{(i)}(t) = 1 + u_i(t)$ with $u_i(t) = \sigma\left(k_i \sum_{j\in\mathcal{N}_i} a_{ij}(P_j(t) - P_i(t))\right)$ bounded in $[-\delta, \delta]$, where k_i are appropriate control gains, the playback times will be synchronized asymptotically.*

Remark 1. When consensus is reached at a playing time $t+\alpha$ ($\alpha < 0$), all clients will watch the live streaming content with a temporal delay of α with respect to the current event time transmitted by the video provider.

Remark 2. Plugging (9) into (3) for a generic user i under Assumption 1, we obtain the following dynamics:

$$\dot{q}_i(t) = f_i(t) - p_{r,i}(t) = (f_i(t) - u_i(t)) - 1 \tag{20}$$

As a consequence, $u_i(t)$ can be seen as a disturbance in the filling rate that the ABR algorithm is able to reject in order to avoid a rebuffering event.

Theorem 4. *Consider a strongly connected digraph G to which a leader node is added imposing a constant state x_0. If it is possible to define a spanning tree in the new graph with the leader as its root, then the system (18) augmented with the leader node achieves leader-follower consensus under the control strategy (15).*

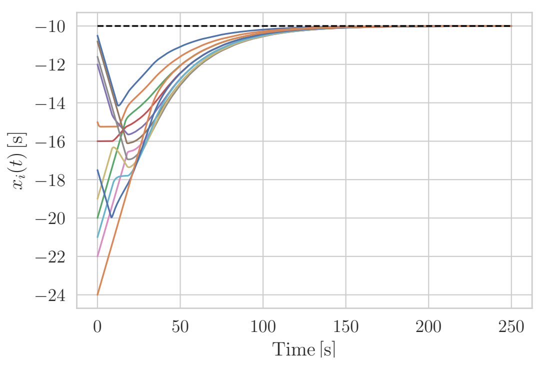

Fig. 5. States dynamics $x_i(t)$ for the ring topology with a leader node

The proofs of Theorem 2, Corollary 3 and Theorem 4, following from [14], [15], are omitted due to space constraints.

IV. Results

In this Section, we consider some network topologies to show the effectiveness of our approach. Once t_s is fixed, i.e., a time instant when all the clients are playing the video content, we define the initial states of the nodes identifying the initial time delays T_{0i} that each client experiences. We have set $k_i = 1$ and $u_i(t) \in [-\delta, \delta]$, $\delta = 0.3$, $\forall\, i \in \mathcal{V}$. In the following simulations, we suppose that each node sends information to and receives information from all of its neighbours. Notice that this assumption could also be removed as long as the graph stays strongly connected. As a first example, let us consider a directed strongly connected ring topology with $N = 13$ nodes. In this topology, each node can communicate only with two neighbours.

As we can see from Figure 3, consensus is reached at roughly -17 seconds, which means that $P_i(t) = t - 17$, $\forall\, i \in \mathcal{V}$, thus implying that all the clients are synchronizing around a playout time delayed of 17 seconds. Notice that in the simulations the zero corresponds to the chosen t_s.

In Figure 4 we consider the leader-following approach for the ring topology by adding a fourteenth leader node (green node in the figure) imposing a state equal to -10, which guarantees a reduced temporal delay.

Let us assume that the leader node can communicate only to one other node. As expected, a consensus is reached at -10 (Figure 5) after about 150s. Notice that the dashed black line in the figure represents the state set by the leader node. Moreover, the transient can be made smaller if the leader is allowed to communicate also with other nodes or by properly increasing the gains k_i. Once the consensus is achieved, it will result that $P_i(t) \simeq P_j(t) \simeq t - 10\ \forall\, i, j = 1, \ldots, N$, which implies that all the clients are synchronized with a delay lower than the leaderless case (Figure 3). Figure 6 shows the dynamics of the control inputs. In particular, after a transient in which most of the control variables experience a saturation, they converge to zero as desired.

Let us now consider a different topology of the network composed of three groups of clients. Each client in a group is

Authorized licensed use limited to: Politecnico di Bari. Downloaded on December 13,2023 at 10:29:52 UTC from IEEE Xplore. Restrictions apply.

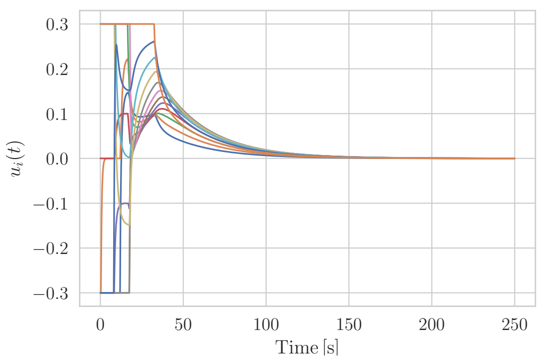

Fig. 6. Control inputs $u_i(t)$ in the case of a ring topology with a leader node

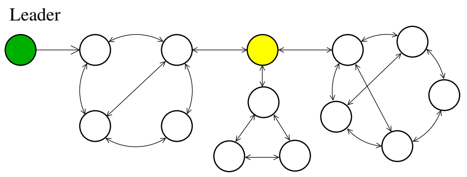

Fig. 7. Network topology with groups of clients and a leader

connected to the others through several edges, while groups are connected with fewer edges as depicted in Figure 7. Notice that this graph, as the previous cases, contains 13 nodes with the same initial states and control input bounds. Also in this case, we consider a fourteenth leader node (green node in the figure) influencing only one other node in the network with the purpose of making the states converge to -10. In this case, the transient time needed to reach consensus is greater than 200 seconds (Figure 8) due to the different network topology and the specific node the leader influences. In fact, if we suppose the leader communicates with the yellow node instead, the time required to obtain consensus with input saturation constraints considerably decreases, up to less than 100 seconds. It is important to point out that the number of available video segments encoded by the DASH standard is 5-10. Since the typical segment duration is $\tau = 5\,\mathrm{s}$, the clients playback time can be delayed with respect to the video provider of an amount that does not exceed 50 seconds. Then, if the leader communicates with several nodes, the synchronization time should be always acceptable for live streaming events of generally long duration.

V. Conclusions

In this paper, we have proposed a distributed control approach to synchronize clients watching live streaming content in geographically distributed locations. We have shown that the playback synchronization problem can be formulated as a consensus problem of integrators. Simulations on different network topologies prove the effectiveness of our approach independently of the protocols adopted. Our future work will focus on the experimental evaluation of the proposed approach on real live streaming scenarios.

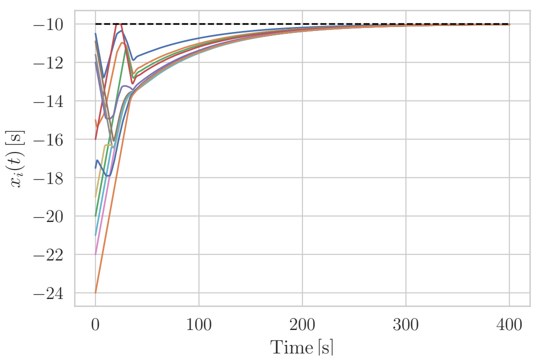

Fig. 8. States dynamics $x_i(t)$ for the topology of Fig. 7

References

[1] G. Cofano, L. De Cicco, and S. Mascolo, "Modeling and design of adaptive video streaming control systems," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 1, pp. 548–559, 2016.

[2] T. Stockhammer, "Dynamic adaptive streaming over http– standards and design principles," in *Proc. of ACM Multimedia Systems (MMSys)*, 2011, pp. 133–144.

[3] L. De Cicco, V. Caldaralo, V. Palmisano, and S. Mascolo, "ELASTIC: a client-side controller for dynamic adaptive streaming over HTTP (DASH)," in *Proc. of Packet Video Workshop*, 2013, pp. 1–8.

[4] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over http," in *Proc. of ACM Conference on Special Interest Group on Data Communication (SIGCOMM)*, 2015, pp. 325–338.

[5] D. Pauwels, J. van der Hooft, S. Petrangeli, T. Wauters, D. De Vleeschauwer, and F. De Turck, "A web-based framework for fast synchronization of live video players," in *Proc. of IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2017, pp. 524–530.

[6] D. Marfil, F. Boronat, A. Sapena, and A. Vidal, "Synchronization mechanisms for multi-user and multi-device hybrid broadcast and broadband distributed scenarios," *IEEE Access*, vol. 7, pp. 605–624, 2018.

[7] M. Montagud, F. Boronat, H. Stokking, and R. van Brandenburg, "Inter-destination multimedia synchronization: schemes, use cases and standardization," *Multimedia systems*, vol. 18, no. 6, pp. 459–482, 2012.

[8] B. Rainer and C. Timmerer, "Self-organized inter-destination multimedia synchronization for adaptive media streaming," in *Proc. of ACM Multimedia (MM)*, 2014, pp. 327–336.

[9] L. Dal Col, S. Tarbouriech, L. Zaccarian, and M. Kieffer, "A linear consensus approach to quality-fair video delivery," in *53rd IEEE Conference on Decision and Control*. IEEE, 2014, pp. 5296–5301.

[10] D. Geerts, I. Vaishnavi, R. Mekuria, O. Van Deventer, and P. Cesar, "Are we in sync? synchronization requirements for watching online video together." in *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*, 2011, pp. 311–314.

[11] B. Rainer and C. Timmerer, "Adaptive media playout for inter-destination media synchronization," in *Proc. International Workshop on Quality of Multimedia Experience (QoMEX)*, 2013, pp. 44–45.

[12] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on automatic control*, vol. 49, no. 9, pp. 1520–1533, 2004.

[13] H. Zhang, Z. Li, Z. Qu, and F. L. Lewis, "On constructing Lyapunov functions for multi-agent systems," *Automatica*, vol. 58, pp. 39–42, 2015.

[14] J. Fu, G. Wen, T. Huang, and Z. Duan, "Consensus of multi-agent systems with heterogeneous input saturation levels," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 66, no. 6, pp. 1053–1057, 2018.

[15] Y. Xie and Z. Lin, "Global optimal consensus for multi-agent systems with bounded controls," *Systems & Control Letters*, vol. 102, pp. 104–111, 2017.

Authorized licensed use limited to: Politecnico di Bari. Downloaded on December 13,2023 at 10:29:52 UTC from IEEE Xplore. Restrictions apply.