Live Streaming Synchronization Using Event-triggered Consensus Control

Gioacchino Manfredi, Vito Andrea Racanelli, Luca De Cicco, Saverio Mascolo

***Abstract*— The advent of social media applications and mobile devices have allowed users to experience live streaming events (e.g. a football match) together, even if they are not in the same physical place. However, this service brings along the issue of ensuring a synchronized video playback among geographically distributed users. Users leave comments and reactions to an online live event on social networks. As a consequence, an unsynchronized video playback can be easily noticed and be detrimental to users' feelings of togetherness. In this work, we propose a distributed control approach to achieve synchronization among users. In particular, the well-known consensus problem of first-order integrators with saturated inputs is employed to design a distributed playback synchronization framework. Furthermore, we propose a leader-follower approach to ensure a controlled synchronization among users in order to obtain the least possible delay with respect to the video content provider. Finally, an event-triggered control is introduced as an enhancement to the previously developed control with the aim of reducing the information exchanged among users. Simulations on different network topologies confirm that the proposed approach is effective at enforcing asymptotic synchronization.**

***Index Terms*— Video Streaming; Consensus; multi-agent systems; event-triggered control.**

I. Introduction and background

Video streaming services are increasingly replacing classical TV broadcast channels since they offer a wide range of contents both on-demand (e.g., movies, TV series) and live (e.g., sport, news). Unlike traditional TV channels, online services allow providers to gather a considerable amount of real-time feedback information on viewers' behaviour and preferences. In this way, providers are able to efficiently personalize advertisements and recommendations.

Nevertheless, video streaming services present two main issues that negatively affect the user's Quality of Experience (QoE). First, video playback might stall or video quality might degrade impacting *service smoothness* and second, in live events, geographically distributed users might watch the same content but with different playback times having detrimental effects on the *service togetherness*, which is the level of satisfaction that users perceive when feeling that the service is experienced together with a number of users (f.i., friends). Media content providers must design their services to maximize the user's perceived quality in an effort to reduce service abandonment. Service smoothness and QoE are an issue on the Internet because network resources, e.g. bandwidth or buffers, are shared and can get congested. On the contrary, traditional broadcast services deliver content via not shared *dedicated medium* so that the quality of the service can be guaranteed more easily.

Video streaming content is delivered to clients through an Internet path whose network bandwidth is unpredictable and time varying. When the video encoding bitrate is larger than the available network bandwidth, the video playback will eventually stall due to playout buffer depletion [1]. For this reason, according to the standard employed for media streaming content, i.e., the *MPEG-Dynamic Adaptive Streaming over HTTP* (MPEG-DASH or DASH) [2], the video bitrate can be adapted to the available network bandwidth. DASH requires that each video is divided into *segments* or *chunks* of fixed duration τ (typically from 1 to 10 seconds). Then, each segment is compressed to create a number of *representations*, or *levels*, to which different encoding bitrates and video resolutions are associated. The DASH standard allows clients' players to dynamically choose the video level that matches the available network bandwidth using an Adaptive BitRate (Adaptive BitRate (ABR)) algorithm. The goal of the ABR algorithm is to maximize the overall quality perceived by the client given the Internet available bandwidth. As a consequence, such control algorithms are designed to avoid as much as possible rebuffering events occurring when the playout buffer gets depleted and video reproduction is interrupted [3].

ABR control algorithms have been extensively studied in the literature [4], [5], [1]. On the other hand, the issue of synchronizing video streams to offer service *togetherness* when users in different locations watch together a live video content is still unsolved.

Consider a group of people geographically distributed in different locations and concurrently watching a live event (e.g. a football match). If streams are not synchronized among users, some of them may watch a particular event (e.g. a goal is scored in a football match) before the others, thus

This work was partially supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART", CUP: D93C22000910001), partially funded by the "LOREN" Project n. 20223Y85JN under the PRIN (Progetti di Ricerca d'Interesse Nazionale) 2022 of the Italian Ministry of Research, and partially funded by the "ELENA" Project n. 13910 under the Next Generation EU Italian National Recovery and Resilience Plan (NRRP) - Bando MedITech 2023 n.3.

Gioacchino Manfredi, Vito Andrea Racanelli, 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: name.surname@poliba.it

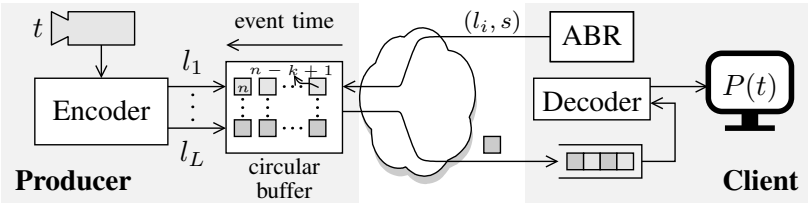

Fig. 1: Live video streaming system

negatively impacting users' feelings of togetherness. As it will be explained later, synchronization issues arise mainly due to the different time instants users join a live streaming event.

When a user starts watching a live streaming event, the ABR algorithm starts fetching the chunks that have been produced most recently by the content provider. In order to mitigate stalls, ABR algorithms are required to fetch a suitable number of segments and place them in the playout buffer before the playback can be started. Thus, the time instant the video starts being reproduced depends on the ABR algorithm controller's parameters and on the network bandwidth availability.

The published literature offers few works, all of them advocating web-based solutions using Real-Time Transport Protocol (RTP) or HTTP Adaptive Streaming (HAS). For instance, some works adopt different approaches for DASH in which all the nodes have to synchronize through the exchange of specific Media Presentation Description (MPD) files. Most of such works employ centralized approaches [6], [7], whose drawback is that they do not scale with the number of users since they require a server to handle synchronization messages form/to possibly millions of nodes. In this context, several standards have been conceived for web-based media synchronization [8]. However, works proposing a decentralized approach resort to heuristics that do not allow a rigorous and systematic analysis of system properties [9], [10].

In this paper, we consider the problem from a rigorous control-theoretic perspective, whose main advantage is to be context-independent, i.e., the control laws can be implemented independently of the protocols and frameworks employed in real-world video delivery systems. As such, no modifications to the protocols employed are required. In particular, we propose a fully distributed approach for video playout synchronization in which each client in the network can communicate/receive the playback time only to/from a limited set of neighbors. Thus, clients do not require centralized information from servers, i.e. the absolute playback time.

The main contributions of this paper can be summarized as follows:

- a mathematical model of the playback time of an event that clearly explains the causes for synchronization issues is provided (Section II);
- the problem of synchronizing video players is stated as a consensus problem involving integrators with saturated control inputs (Section III);
- the absolute delay of users with respect to the video content provider is reduced by adopting a leader-follower approach (Section III);
- an event-triggered control avoids users continuously sending information to their neighbours so that communication occurs only at specific time instants, which reflects realistic cases and still guarantees leader-follower consensus achievement (Section IV).

As a matter of fact, this work significantly extends [11] by (i) providing a more detailed explanation of live video streaming systems and thoroughly describing the problem of synchronizing users' playback times; (ii) proving theorems and corollaries stating the achievement of global asymptotic consensus in the case of a live streaming event; (iii) designing an event-triggered control to avoid a continuous exchange of information among users and, when the lack of synchrony between all agents is low, imposing a stopping criterion that prevents the agents from sending messages indefinitely; (iv) expanding the results to show the convergence of the proposed synchronization mechanisms when event-triggered control is employed.

To the best of our knowledge, this is the first work to tackle the synchronization issue of live streaming events from a control theoretic perspective. Furthermore, an important design choice in our solution is that the synchronization algorithm is decoupled from the ABR algorithm. This approach has the merit of making the deployment of our solution very easy as it does not require to make any change in the ABR algorithm.

The control input leveraged to synchronize clients is the playback rate, an approach denoted as *Adaptive Media Playout (AMP)* in the literature [12], [13]. In practice, the playback rate, i.e. the speed of reproduction of the video, *can be slightly adjusted* to control the playback time. This technique has been used also to enhance the client's buffer memory requirements and to reduce playout interruptions [14], [15]. It is important to point out that issues related to audio streaming with AMP have been widely addressed [16], [17], [18], and therefore will not be accounted for in this work. Furthermore, several studies on the user's perceived quality, or *Quality of Experience (QoE)*, show that the playback rate can be increased/decreased by only a small amount to prevent the QoE from being negatively impacted [19], [20]. Thus, to limit QoE degradation, our approach considers the playback rate (i.e., the control input) to be bounded in a given interval. Under these conditions, we show that our approach allows asymptotic convergence of all the clients to the same playback time (Section III) also in the case of event-triggered control (Section IV).

II. Playback Time Model

In Section II-A we provide a model of the playback time. Based on this model, in Section II-B we show how the de-synchronization arises among video players. Finally, in Section II-C we present a model of the Adaptive Media Playout approach to be used to synchronize players.

A. Video playback time model

In this section, we provide a model of the playback time of a user watching a live event that starts at time $t = 0$. We assume that the time of the live event is the same as the real time t, therefore the term *time* end *event time* will be used interchangeably in this paper.

Figure 1 shows how a DASH-compliant live streaming system, such as YouTube or Facebook, produces live content. A camera captures a live scene that is compressed in real-time by an encoder. The latter produces video segments each τ seconds by encoding the same video content at different bitrate levels (or resolutions) l_i belonging to a discrete set $\mathscr{L} = \{l_1, \ldots, l_M\}$ ($l_i < l_{i+1}$). For each level $l_i \in \mathscr{L}$, the *circular buffer* depicted in the figure receives and stores the last k segments produced by the encoder. A circular buffer is a fixed-size buffer where the starting location chosen to store the first segment is not important and the extraction of data follows a *first in, first out* (FIFO) logic. Therefore, clients will start downloading the oldest segments in the buffer. In the same way, when the circular buffer is full and a new chunk has to be stored, the oldest segment is overwritten. In other words, this buffer contains only the most recently produced k segments, which are made available for download to the clients that want to join the live streaming event. Each segment of duration τ is present in the circular buffer with M different levels of resolution. The circular buffer contains a number of segments representing the scene in a time window $W(t)$, as defined in the following. Let $s(t)$ be the segment index that contains the video scene at time t:

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

where $\lfloor \cdot \rfloor$ is the *floor* operator that maps a real number x to the greatest natural number less than x. According to its definition, the segment $s(t)$ contains the event scene in the time interval $[s(t)\tau, (s(t)+1)\tau[$. Hence, at time t, the circular buffer of k segments contains video scenes in the time window:

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

where $n = s(t)$ is the n-th chunk.

Let t_J be the time at which a user joins the live event. In t_J the user immediately starts downloading the oldest segment available in the circular buffer, whose index is $s(t_J) - k$, since the buffer contains k segments. Once the segments are downloaded, they are temporarily stored in the client's playout buffer whose level, measured in seconds, is denoted with $q(t)$. As already explained, for each segment to be delivered, the ABR control algorithm selects the video level $l(t) \in \mathscr{L}$. In general, before starting the playback of the video, an ABR control algorithm fills the playout buffer until the level $q(t)$ reaches a minimum value, say q_L, that is considered enough to mitigate the rebuffering events occurring when the buffer gets completely depleted and video reproduction is interrupted ($q(t) = 0$). 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 video fill rate in seconds of video for unit of second, 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 other words, the playback rate is the speed at which the video is played on the user's screen.

The video fill rate $f(t)$ can be modelled as follows:

$$f(t) = \frac{\Delta T_{\text{video}}}{\Delta T} = \frac{\Delta data}{\Delta T} \cdot \frac{\Delta T_{\text{video}}}{\Delta data} = \frac{r(t)}{l(t)} \tag{4}$$

where $r(t) = \Delta data / \Delta T$ is the download rate, which depends on the time-varying network bandwidth measured in bytes/s, and $l(t) = \Delta data / \Delta T_{\text{video}}$ is the encoding quality level [1].

In normal conditions, the playback rate $p(t)$ is equal to 1 when the video is playing and 0 when the video is paused. Thus, the following holds:

$$p(t) = \begin{cases} 1 & \text{playing} \\ 0 & \text{paused} \end{cases} \tag{5}$$

Let us now model the playback time $T(t)$, which is the time instant of the video that the user is watching at time t. Notice that in general the playback time $T(t)$ is different from the time t of the live event because of the unavoidable delays due to encoding, decoding and propagation. Once the user joins the live event at time $t = t_J$, the playback time of the first segment downloaded is:

$$T_0 = s(t_J)\tau - k\tau \tag{6}$$

where T_0 corresponds to the initial time of the first segment stored in the producer's circular buffer at time t_J. Notice that the video is not played at the client ($p(t) = 0$) until the queue level $q(t)$ is higher than 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 $q(t_B) = q_L$. Therefore, t_B is influenced both by the download rate $r(t)$, which can be considered as a disturbance, and by $l(t)$ that is chosen by the ABR control algorithm.

For these considerations, the reproduction of the video at the player starts at time $t_P = t_J + t_B$. Thus, when $t \geq t_P$, the playback time dynamics is given by:

$$\dot{T}(t) = p(t) \tag{7}$$

with initial condition $T(t_P) = T_0$. By considering that the reproduction of the video starts at t_P and by supposing no rebuffering events occur after t_P, i.e., $p(t) = 0$ for $t < t_P$ and $p(t) = 1$ for $t \geq t_P$, we can integrate (7) between t_p and a generic time instant t taking into account the initial condition:

$$\begin{aligned} T(t) = \int_{-\infty}^{t} p(\xi)d\xi = \int_{-\infty}^{t_p} p(\xi)d\xi + \int_{t_p}^{t} p(\xi)d\xi = \\ = T_0 + \int_{t_p}^{t} p(\xi)d\xi = T_0 + t - t_P \end{aligned} \tag{8}$$

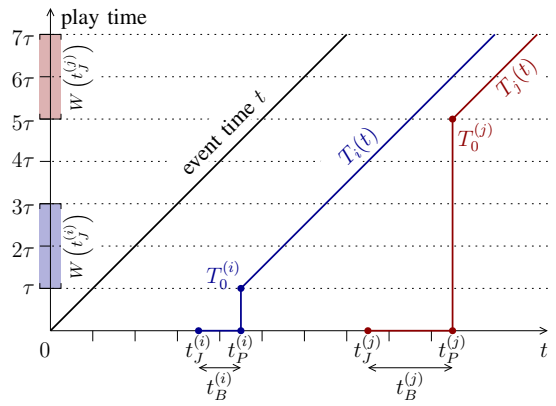

Fig. 2: Playback time dynamics of two users i and j joining at different time instants $t_J^{(i)}$ and $t_J^{(j)}$

B. The synchronization issue

Figure 2 is useful to explain the issue of synchronization. The figure depicts the playback time of two users, say i and j, that are interested in watching the same live streaming event. In this example, a live video streaming system is considered with a circular buffer containing the two most recently produced segments, i.e., $k = 2$. As one can see in the figure, 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, which are those contained in the time window $W(t_J^{(i)})$ (see the blue shaded area in the figure). The buffer time length $q(t)$ reaches q_L after $t_B^{(i)}$ seconds so that at $t_P^{(i)} = t_J^{(i)} + t_B^{(i)}$ the playback on the user's device starts, i.e. $p(t) = 1$ for $t \geq t_P^{(i)}$. Moreover, the playback starts from a play time $T_0^{(i)}$, as defined in (6), which is the play time of the video at the beginning of the first chunk stored in the buffer, i.e. at the beginning of the window $W(t_J^{(i)})$. At time $t_J^{(j)} > t_J^{(i)}$, user j also joins the live event. Analogously, the ABR algorithm retrieves the video chunks identified by the time window $W(t_J^{(j)})$ (see red shaded area in the figure) and stores them in the playout buffer until it reaches the minimum level q_L needed to start playing the video on the user's device[1]. Morover, the time $t_B^{(j)}$ needed to reach q_L in the playout buffer of user j is different in general from $t_B^{(i)}$. Therefore, user j will start watching the event at time $t_P^{(j)} = t_J^{(j)} + t_B^{(j)}$ from a play time $T_0^{(j)}$. As computed in (8), the playback time of user i is $T_i(t) = T_0^{(i)} + t - t_P^{(i)}$ while the playback time of user j is $T_j(t) = T_0^{(j)} + t - t_P^{(j)}$. Hence, the lack of synchrony between the two users is equal to:

$$T_i(t) - T_j(t) = [(T_0^{(i)} - t_J^{(i)}) - (T_0^{(j)} - t_J^{(j)})] + (t_B^{(j)} - t_B^{(i)}) \quad (9)$$

where $T_0^{(i)} - t_J^{(i)}$ is the distance between $t_J^{(i)}$ and the play time $T_0^{(i)}$ of the video at the beginning of the first chunk stored in the buffer. The same can be said for $T_0^{(j)} - t_J^{(j)}$ in the case of user j. The two users are synchronized when $T_i(t) - T_j(t) = 0$.

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

Based on the above, we can conclude that the lack of synchrony between two users depends on their join and buffering times.

C. The Adaptive Media Playout model

We have shown that if the video playback rate of players is constant, playback times are in general not synchronized in accordance to (9). In this section, we present the AMP approach and show how it can be leveraged to synchronize users. The AMP approach consists of slightly varying the playback rate around the nominal value 1, i.e. the playback must be slightly slowed down or sped up of a time-dependent amount, say $u(t)$. Clearly, $u(t)$ must be small enough to be barely noticeable by the user so that the user's QoE is not affected [19], [20].

Let us redefine the playback rate by adding this term:

$$p_r(t) = p(t)(1 + u(t)) \quad (10)$$

where $u(t) \in [-\delta, \delta]$, with δ small enough, and $p(t)$ is the playback rate (5). Thus, the playback time now becomes:

$$T(t) = \int_{t_P}^{t} p(\xi)(1 + u(\xi))d\xi \quad \forall\, t \geq t_P \quad (11)$$

with initial condition $T(t_P) = T_0$. 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

$$T_i(t) = \int_{t_P^{(i)}}^{t} (1 + u_i(\xi))d\xi \quad (12)$$

whose initial condition is $T_i(t_P^{(i)}) = T_0^{(i)}$. The playback rate variations $u_i(t)$, $i = 1, ..., N$, are the control variables that will enforce synchronization among users. The synchronization can be formally stated as: $e_{ij}(t) = T_i(t) - T_j(t) \to 0\ \forall i, j = 1, ..., N$, $i \neq j$, exponentially.

III. The Proposed Synchronization Approach

Based on the model of the playback time for each client consuming the same live streaming content, this section presents the proposed distributed approach to achieve synchronization through the formulation of a consensus problem involving integrators and saturated inputs.

The design requirements are the following: (R1) the system must be horizontally scalable, i.e., it should work also in the case of large events (N large); (R2) it has to be implementable using technologies already available in the media distribution industry.

To meet the design criteria (R1), we propose to use a decentralized control approach, which involves messages to be sent only among selected users. To exchange synchronization messages directly among users, without the need of a central server, the WebRTC open standard could be used, which is a widely available technology that allows real-time

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

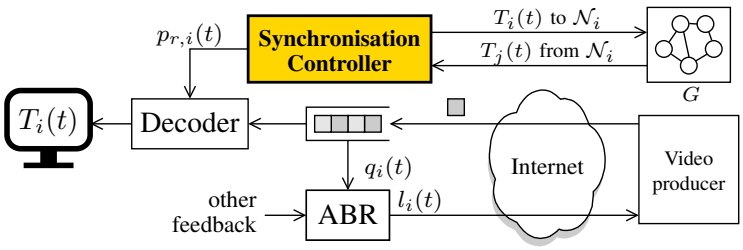

Fig. 3: The proposed synchronization control architecture

communication among browsers and is supported by all major Internet browsers. To meet the requirement (R2), the video is delivered using the standard DASH protocol specification without requiring any change to the ABR algorithm running at the client.

Figure 3 shows the architecture of the proposed approach for playback time synchronization. The i-th client runs an ABR algorithm that, based on information such as the estimated available bandwidth and the playout buffer level $q_i(t)$, dynamically selects the chunk bitrate level $l_i(t)$ and requests it to the video producer server [1], [4]. The server delivers the video segments encoded at the level required by the ABR algorithm. When the segments are delivered, the client stores them in the playout buffer to play them. The *synchronization controller*, which sends to and receives information from the neighboring devices, is the core of the proposed algorithm. Such a controller selects the most suitable value of the playback rate $p_{r,i}(t)$ at which the video has to be played at client i. As mentioned above, the value of $p_{r,i}(t) = 1 + u_i(t)$ should stay as close as possible to 1 to mitigate QoE degradation. Thus, we require $u_i(t)$ to be bounded in a set $[-\delta, \delta]$ and to converge to zero when synchronization is achieved. Finally, the decoder decompresses the video frames drained from 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 for synchronization, let us introduce the following non-restrictive assumption: *Assumption 1.* Once each user $i = 1, \dots, N$ starts the playback of the live streaming content, no rebuffering event occurs, which implies (i) $p_i(t) = 1 \; \forall t \geq t_P^{(i)}$ and (ii) the ABR algorithm is able to avoid playout buffer depletion, i.e. $q_i(t) > 0, \forall t \geq t_P^{(i)}$.

The proposed approach is based on the dynamic model of the playing time evolution, which is derived from (12).

Let us start by defining the initial delay of the i-th user as $T_{0i} = T_0^{(i)} - t_P^{(i)} < 0$. The synchronization algorithm will start at time $t = t_s$ when all the N users attending the event have started the video playback, i.e., $p_i(t) = 1 \, \forall t > t_s, \forall i \in \{1, \dots, N\}$. Since for $t < t_s$ the synchronization algorithm is not active, it must result that $u_i(t) = 0, \forall i \in \{1, \dots, N\}$. As a consequence, the playback time can be obtained as follows:

$$T_i(t) = T_{0i} + t + \int_{t_P^{(i)}}^{t} u_i(\xi) d\xi = T_{0i} + t + \int_{t_s}^{t} u_i(\xi) d\xi \quad (13)$$

$\forall t \geq t_s$, 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 of $T_i(t)$ is:

$$\dot{T}_i(t) = 1 + u_i(t) \quad (14)$$

Next, we perform a change of coordinates to obtain a set of integrators: let $x_i(t) = T_i(t) - t$, then $\dot{x}_i(t) = \dot{T}_i(t) - 1 = u_i(t), \; \forall t \geq t_s$. As a consequence, controlling the models of all the clients is equivalent to controlling a set of integrators as defined in the following:

$$\begin{cases} \dot{T}_1(t) = 1 + u_1(t) \\ \vdots \\ \dot{T}_N(t) = 1 + u_N(t) \end{cases} \equiv \begin{cases} \dot{x}_1(t) = u_1(t) \\ \vdots \\ \dot{x}_N(t) = u_N(t) \end{cases} \quad (15)$$

The physical meaning of the variable $x_i(t)$ is the temporal delay of user i with respect to the current playout time transmitted by the video provider. We are now ready to model the stated problem as a consensus problem. To this end, let us define a directed graph (or digraph) $G(\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \dots, v_N\}$ is the set of nodes, also identified simply by its indices $i = 1, \dots, N$, 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 set of neighbors of node v_i is defined as $\mathcal{N}_i = \{v_j \in \mathcal{V} : (v_j, v_i) \in \mathcal{E}\}$.

At this point, one could model the clients as the nodes $i \in \mathcal{V}$ of the graph G, where each node i is described by the corresponding state $x_i(t)$ and receives information from a certain number of neighbours $\mathcal{N}_i$.

Supposing the control inputs $u_i(t)$ are unbounded, then for a set of integrators we can employ the following well-known control strategy [21]:

$$u_i(t) = \sum_{j \in \mathcal{N}_i} a_{ij}(x_j - x_i) = \sum_{j \in \mathcal{N}_i} a_{ij}(T_j - T_i) \quad (16)$$

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

The resulting system is

$$\dot{\boldsymbol{x}}(t) = -L\boldsymbol{x}(t) \quad \forall t \geq t_s \quad (17)$$

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 the adjacency matrix A. Let $D = D(G) = (d_{ij})$ be the in-degree matrix with $d_{ij} = in-deg(v_i)$ if $i = j$ and $d_{ij} = 0$ otherwise, where $in-deg(v_i)$ is the in-degree of node v_i, i.e. the number of edges pointing towards node v_i. Then, the Laplacian matrix can be defined as $L = D - A$. 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 is that the control protocol (16) solves the consensus problem for the set of integrators (15) globally and asymptotically [21]. Therefore, it can be stated that (17) is stable and there exists an $\alpha \in \mathbb{R}$ s.t. the system converges to the equilibrium point $\bar{\boldsymbol{x}} = \alpha \mathbf{1}$, i.e., $\bar{x}_i = \alpha, \forall\, i \in \mathcal{V}$. In addition, if G is balanced, i.e., the in-degree is equal to the out-degree for each node, then $\alpha = \mathbb{E}[x_i(t_s)] = \frac{1}{N}\sum_{i=1}^{N} x_i(t_s)$.

When consensus is achieved, $x_i(t) = T_i(t) - t = T_{0i} + \int_{t_s}^{t} u_i(\xi)d\xi = \alpha$ and thus $T_i(t) = t + \alpha$ for all i. In other words, if the consensus problem is solved, then also the playback time synchronization problem is solved. Notice that when G is balanced $\alpha = \mathbb{E}[x_i(t_s)]$, so it turns out that $T_i(t)$ converges to $t + \mathbb{E}[T_{0i}]$, where the second term represents the average of the initial delays of the clients.

However, in the previous approach, unbounded control inputs $u_i(t)$ were considered, which is not a realistic case. As already explained in Section II-C, all the control inputs should be bounded, i.e., $|u_i(t)| \leq \delta\ \forall\, i \in \mathcal{V}$ to avoid a perceptible speedup or slowdown of the video, which would bring to a QoE deterioration. To tackle such an issue, let us introduce the following saturation function:

$$\sigma(x) = \begin{cases} \delta & x > \delta \\ x & -\delta \leq x \leq \delta \\ -\delta & x < -\delta \end{cases} \tag{18}$$

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

Therefore, to guarantee that $|u_i(t)| \leq \delta,\ \forall\, i \in \mathcal{V}$, we can write:

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

If we employ the same control strategy as in (16), we obtain $\dot{x}_i(t) = \sigma(u_i(t)) = \sigma\left(k_i \sum_{j\in\mathcal{N}_i} a_{ij}(x_j - x_i)\right)$, where $k_i > 0,\ \forall\, i \in \mathcal{V}$ are controller gains. Therefore, it results that

$$\dot{\boldsymbol{x}} = \sigma(-KL\boldsymbol{x}) \tag{20}$$

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

Before introducing the main result, we recall the following lemma from [22]:

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*

$$R = \text{diag}\{r_1, \ldots, r_N\},\ Q = RL + L^\top R \tag{21}$$

Then $R > 0$, $Q \geq 0$ and the kernel of Q has dimension 1 and is given by $\text{span}\{\mathbf{1}_N\}$.

Let us also recall the following Corollary of LaSalle theorem from [23]:

Corollary 2. *Let $\boldsymbol{x} = \mathbf{0}$ be an equilibrium point for the system $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$. Let $V : \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable, radially unbounded, positive definite function such that $\dot{V}(\boldsymbol{x}) \leq 0$ for all $\boldsymbol{x} \in \mathbb{R}^n$. Let $\mathcal{S} = \{\boldsymbol{x} \in \mathbb{R}^n | \dot{V}(\boldsymbol{x}) = 0\}$ and suppose that no solution can stay identically in $\mathcal{S}$ other than the trivial solution $\boldsymbol{x}(t) \equiv 0$. Then, the origin is globally asymptotically stable.*

We are now ready to prove the following:

Theorem 3. *Consider a multi-agent system represented by a graph G whose dynamics are described by (19). Suppose each agent applies the control (16) and assume that G is a strongly connected and directed graph. Then the control (16) globally asymptotically solves a consensus problem.*

Proof: Consider (20) and let $\boldsymbol{z} = -L\boldsymbol{x}$. Then, we can define the following dynamics:

$$\dot{\boldsymbol{z}} = -L\dot{x} = -L\sigma(K\boldsymbol{z}) \tag{22}$$

In order to prove the convergence to 0 of such a system, consider the Lyapunov candidate function [24], [25]:

$$V(\boldsymbol{z}) = \sum_{i=1}^{N} \int_0^{z_i} r_i \sigma(k_i q) dq \tag{23}$$

where r_i is the i-th element of the vector $\boldsymbol{r}$ (Lemma 1). It can be observed that $V(\boldsymbol{z}) > 0\ \forall\, \boldsymbol{z} \neq \mathbf{0}$, $V(\mathbf{0}) = 0$ and $V(\boldsymbol{z})$ is radially unbounded, i.e., $V(\boldsymbol{z}) \to +\infty$ as $\|\boldsymbol{z}\| \to +\infty$. Applying the fundamental theorem of calculus, the derivative of (23) yields:

$$\begin{aligned} \dot{V}(\boldsymbol{z}) &= \sum_{i=1}^{N} \frac{dV}{dz_i}\frac{dz_i}{dt} = \sum_{i=1}^{N} r_i \sigma(k_i z_i)\dot{z}_i = \sigma(K\boldsymbol{z})^\top R\dot{\boldsymbol{z}} = \\ &= -\sigma(K\boldsymbol{z})^\top RL\sigma(K\boldsymbol{z}) \end{aligned} \tag{24}$$

Given any $A \in \mathbb{R}^{n\times n}$ and $\boldsymbol{x} \in \mathbb{R}^n$, it is well-known that $\boldsymbol{x}^\top(A + A^\top)\boldsymbol{x} = \boldsymbol{x}^\top A\boldsymbol{x} + \boldsymbol{x}^\top A^\top \boldsymbol{x} = 2(\boldsymbol{x}^\top A\boldsymbol{x})$. Then, recalling that R is symmetric and that $Q = RL + L^\top R$, it follows that

$$\begin{aligned} \dot{V}(\boldsymbol{z}) &= -\sigma(K\boldsymbol{z})^\top RL\sigma(K\boldsymbol{z}) = \\ &= -\frac{1}{2}\sigma(K\boldsymbol{z})^\top Q\sigma(K\boldsymbol{z}) \leq 0 \end{aligned} \tag{25}$$

It can be easily seen that $\mathcal{S} = \{\boldsymbol{z} : \dot{V}(\boldsymbol{z}) = 0\} = \{\boldsymbol{z} : \sigma(K\boldsymbol{z}) = a\mathbf{1}_N\}$, with $a \in \mathbb{R}$.

However, if we suppose $a > 0$, then, by definition of the saturation function, it follows that $z_i \geq a/k_i, \forall i \in \mathcal{V}$. Since for Lemma 1 $\boldsymbol{r} > 0$, it follows that if z_i is positive for each i, then we have that $\boldsymbol{r}^\top \boldsymbol{z} = -\boldsymbol{r}^\top L\boldsymbol{x} > 0$, which is a contradiction because, being $\boldsymbol{r}$ a left eigenvector of L, it must result that $\boldsymbol{r}^\top \boldsymbol{z} = 0$. By using the same arguments, we can also rule out the case $a < 0$ and thus the only possible case is $a = 0$. This accounts for the fact that the only solution that can stay identically in $\mathcal{S}$ is the trivial solution $\boldsymbol{z}(t) = \mathbf{0}$. Hence, for Corollary 2, it is possible to state that $\boldsymbol{z}$ globally asymptotically converges to zero.

Notice that $\bar{\boldsymbol{z}} = -L\bar{\boldsymbol{x}} = \mathbf{0}$, where $\bar{\boldsymbol{x}}$ is a right eigenvector of L associated to the zero eigenvalue. Since the dimension

of the eigenspace associated with the zero eigenvalue of L is one, then $\exists\, \alpha \in \mathbb{R}$ s.t. $\bar{\boldsymbol{x}} = \alpha \mathbf{1}_N$. Therefore, it results that $x_i - x_j \to 0$ as $t \to +\infty\ \forall\, i, j$, which concludes the proof.

As a direct consequence of Theorem 3, we can state the following:

Corollary 4. *Consider N users watching the same live streaming event. Suppose the users can receive the playback times $T_j(t)$ 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 $u_i(t) = \sigma\left(k_i \sum_{j\in\mathcal{N}_i} a_{ij}(T_j(t) - T_i(t))\right)$ bounded in $[-\delta, \delta]$, where k_i are appropriate control gains, the playback times will be synchronized asymptotically.*

Proof: Recalling that $x_i(t) = T_i(t) - t$, according to Theorem 3, when consensus is reached, $x_i(t) \to \alpha$ and the playback time $T_i(t) \to t + \alpha$ for all i when $t \to +\infty$, thus implying the achievement of consensus also for the playback time.

Remark 1. When consensus is reached at a playing time $t + \alpha$ ($\alpha < 0$), although all clients are synchronized, they 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 (10) into (3) for a generic user i under Assumption 1, we obtain the following dynamics of the playout buffer:

$$\dot{q}_i(t) = f_i(t) - p_{r,i}(t) = (f_i(t) - u_i(t)) - 1 \tag{26}$$

From the point of view of the ABR algorithm, $u_i(t)$ can be seen as a disturbance in the filling rate. However, the ABR algorithm is able to reject it in order to avoid buffer depletion and therefore a rebuffering event. At steady state, $u_i = 0$, which implies that when consensus is achieved the normal behavior $\dot{q}_i(t) = f_i(t) - 1$ is recovered. At this point, the ABR algorithm, responsible for filling the buffer, no longer depends on the consensus, which is represented by the control variable $u_i(t)$. Hence, the consensus affects the playout buffer dynamics $\dot{q}_i(t)$ but $\dot{q}_i(t)$ does not affect the consensus, and therefore the playback rate, as long as $q_i(t) > 0$, which is always true under Assumption 1.

Theorem 5. *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 (19) augmented with the leader node achieves leader-follower consensus under the control strategy (16).*

Proof: In this setting, the leader node influences–but it is not influenced by–one or more nodes of the graph through directed edges. Let $\bar{G}$ denote the augmented graph including the graph G, the leader node v_0 and all associated edges. Let $A_1 = \mathrm{diag}\{a_{10}, \dots, a_{N0}\}$, where $a_{i0} > 0$ only if there is a directed link from the leader to node i, otherwise $a_{i0} = 0$, and let $U = L + A_1$, with L as defined in *Lemma 1*. In order to continue the proof, we introduce the following [22]:

Lemma 6. *If the augmented graph $\bar{G}$ has a spanning tree with the leader node as the root, then U is full rank. In addition, let*

$$\begin{aligned} \boldsymbol{r} &= [r_1, \dots, r_N]^T = (U^T)^{-1}\mathbf{1}_N, \\ R &= \mathrm{diag}\{r_1, \dots, r_N\}, \\ W &= RU + U^T R, \end{aligned} \tag{27}$$

then, $R > 0$ and $W > 0$.

At this point we set $\hat{x}_i = x_i - x_0$, where x_i is the state associated to the i-th node and x_0 is the state imposed by the leader node. Therefore, we can write $\hat{\boldsymbol{x}} = \boldsymbol{x} - x_0\mathbf{1}_N$, where $\boldsymbol{x}$ follows the dynamics in (19) with the control actions (16). It is easy to verify that $\dot{\hat{\boldsymbol{x}}} = \sigma(-KU\hat{\boldsymbol{x}})$ and that if we define $\boldsymbol{z} = -U\hat{\boldsymbol{x}}$, then $\dot{\boldsymbol{z}} = -U\sigma(K\boldsymbol{z})$. Analogously to the proof of Theorem 3, we consider the candidate Lyapunov function in (23) where now r_i is the i-th element of the vector $\boldsymbol{r}$ defined in (27). Its derivative is

$$\dot{V}(\boldsymbol{z}) = -\frac{1}{2}\sigma(K\boldsymbol{z})^\top W \sigma(K\boldsymbol{z}) \leq 0 \tag{28}$$

The candidate Lyapunov function is radially unbounded and is such that $V(\boldsymbol{z}) > 0\ \forall \boldsymbol{z} \neq \mathbf{0}$ and $V(\boldsymbol{z}) = 0$ when $\boldsymbol{z} = \mathbf{0}$. Moreover, since $W > 0$, then $\dot{V}(\boldsymbol{z}) = 0$ if and only if $\boldsymbol{z} = \mathbf{0}$, otherwise $\dot{V}(\boldsymbol{z}) < 0$. Then, for the Lyapunov stability criterion, $\boldsymbol{z}$ globally asymptotically converges to zero.

Recalling that $\boldsymbol{z} = -U\hat{\boldsymbol{x}}$ and that U is full rank, it follows that $\boldsymbol{z} = \mathbf{0}$ implies $\hat{\boldsymbol{x}} = \mathbf{0}$. Therefore, $x_i \to x_0$ as $t \to +\infty\ \forall i \in \mathcal{V}$.

IV. Distributed event-triggered control

So far we have made the unrealistic assumption that users communicate their playback time to neighbours continuously in time. To reduce the overall need for communication, and therefore the number of messages exchanged among users, we introduce an event-triggered control. In consensus problems, such a methodology, widely studied in the literature [26], [27], [28], consists of updating the control input only when a certain error exceeds a threshold (triggering events). Thus, each agent communicates its state only at specific time instants, called *triggering times*. As a consequence, the control action is piecewise constant since it changes only when triggering times occur. Following this idea, we design an event-triggered linear feedback law for each agent.

Let $\mathcal{T}_i = \{0, t_1^i, t_2^i, \dots, t_{\phi_i}^i, \dots\}$ be the event-triggering time instants for agent i. Then, we can define the following control strategy

$$\hat{u}_i(t) = k_i \sum_{j\in\mathcal{N}_i} a_{ij}(x_j(t_{\Phi_j}^j) - x_i(t_{\Phi_i}^i)) \tag{29}$$

where $\Phi_j \in \mathbb{N}, \forall\, j \in \mathcal{N}_i$, and $\Phi_i \in \mathbb{N}$, denote the fact that different agents have in general different triggering times. Moreover, $t_{\Phi_j}^j \in \mathcal{T}_j$ is such that $t_{\Phi_j}^j = \max\{t_{\phi_j}^j | t_{\phi_j}^j \leq t\}$, where $t_{\phi_j}^j$ is the ϕ_j-th triggering time for agent j. The same can be said for agent i. At this point, we can define the sampled error as $e_i(t) = x_i(t_{\Phi_i}^i) - x_i(t)$, i.e., the difference between the state in the last triggering time instant and the actual value

of the state at the present time t. It can be shown that given the following triggering time update

$$t^i_{\phi_i+1} = \max_{c>t^i_{\phi_i}} \{c | \|e_i(t)\|^2 - \alpha_i e^{-\beta_i t} \leq 0, \forall\, t \in [t^i_{\phi_i}, c]\} \tag{30}$$

with $\alpha_i > 0$ and $\beta_i > 0$, global leader-follower consensus is achieved if and only if the digraph is strongly connected and there exists a spanning tree having the leader node as its root [26].

Then, for each agent $i \in \mathcal{V}$, the system dynamics is

$$\dot{x}_i = \sigma(\hat{u}_i(t)) \tag{31}$$

The value of $\hat{u}_i(t)$ is kept constant until a new triggering time instant is generated as shown in (30). In other words, for agent i, if the error–defined as the difference between the state in the last triggering time instant and the current state–remains below a certain threshold (i.e., $\alpha_i e^{-\beta_i t}$), then the agent will not communicate its state to its neighbours nor will it change its control. This way, whilst the actual state of agent i keeps changing in accordance to (31), the value of the state used to compute the control protocols is considered constant. Therefore, between a triggering time instant and the next one, the agent and its neighbours will use the value of the state in the last triggering time to compute the new value of their control protocol. Only at the new triggering time will the agent update its control protocol and send its state to all of its neighbours so that they can update their own control.

The event-triggered approach bears another practical advantage in the case of leader-follower consensus. When the algorithm begins at time t_s, it is sufficient that the leader shares the reference value with its neighbours only once at t_s. Since each agent keeps the state associated to a neighbour constant until a new value is triggered by the neighbour itself, there is no need for the leader to communicate a second time with its neighbours, which will keep the reference value constant. This considerably reduces the overall communication overhead.

As previously explained, the aforementioned approach allows the agents to communicate with their neighbours only in discrete time instants. It is important to point out that α_i and β_i in (30) are design parameters that regulate the trade-off between messages exchanged and transient. Specifically, reducing communication between agents slows down the achievement of consensus, whereas expediting consensus naturally entails increased message exchange. Furthermore, the analyzed system employing the well-known event-triggered mechanism has been proved to converge without any risk of Zeno behavior [26], thus constraining the frequency of triggers to be limited in a time interval.

However, due to the definition of (30) and in particular to the presence of a decreasing exponential term, the messages exchanged among the agents will never stop. This is also due to numerical issues associated with the computer employed for the simulations. At this point, one could leverage the concept of *finite-time event-triggered consensus* to tackle such a problem. In a nutshell, it consists of proving that consensus is reached after a finite time t^* rather than at infinity. This could be a good solution to our problem since after t^* no triggering event would happen and, consequently, no more information would be exchanged among the agents. Finite-time consensus can be guaranteed by finding a Lyapunov function that satisfies additional and more restrictive conditions.

Several works focus on finite-time event-triggered consensus [29], [30], [31], [32], but, to the best of our knowledge, none of them proves finite-time consensus for the system considered in this work. In particular, finding a Lyapunov function guaranteeing finite-time consensus is not an easy task. For this reason, we consider the following approach: agent i uses the control protocol in (29) when $\exists\, j \in \mathcal{N}_i$ s.t. $|x_j(t^j_{\phi_j}) - x_i(t^i_{\phi_i})| > \gamma$, where $\gamma > 0$ is a small adjustable parameter, and 0 otherwise. This means that once the playback times of the neighbours are close enough to the playback time of agent i, the draining rate of agent i is set back to 1, i.e., $\hat{u}_i(t) = 0$, thus making $x_i(t)$ constant. Since leader-follower consensus is guaranteed according to [26] under the control protocol (29) and the triggering time update given by (30), the approach just described prevents the agents from sending messages indefinitely by stopping the variation of their states and therefore avoiding further triggering times when a *de-facto* consensus is reached. In other words, at steady state, leader-follower consensus is almost achieved since the maximum lack of synchrony between two neighbouring agents is γ, which is small enough not to be noticeable by users. It is worth precising that small communication delays may occur among clients exchanging their current state in a triggering time. This does not hinder convergence since, at each agent, the previous control action is kept constant. Therefore, if the change in the control is slightly delayed, consensus will be achieved anyway but with some additional time. Moreover, in the worst case, the lack of synchrony between the leader node and any other node is at most $N \cdot \gamma$, which is still unnoticeable by users if γ is close to zero. As stated in [19], users start to notice differences in the playback time among them when they are above 500 ms. Hence, $N \cdot \gamma$ should be kept below this threshold.

V. Results

In this Section, we consider some network topologies to show the effectiveness of our approach in baseline cases. Notice that our approach cannot be compared against any other because the literature offers either centralized approaches or context-dependent techniques with different goals. 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 when the server sends the streaming content. We have set $k_i = 1$ and $|u_i(t)| \leq \delta$, $\delta = 0.3$, $\forall\, i \in \mathcal{V}$. It worth noticing that the controller gains $k_i > 0$ can be tuned to impact the synchronization time. In particular, this transient tends to shrink when gains are increased. However, a price is paid in terms of smoothness of the state dynamics and of the control inputs. The latter take more time to converge and the profiles of the controls are less smooth and decay to zero more abruptly. This behaviour is due to the fact that the controller output is subject to a saturation function, therefore setting the controller gains beyond a certain threshold would not bring a significant improvement since the output would saturate and

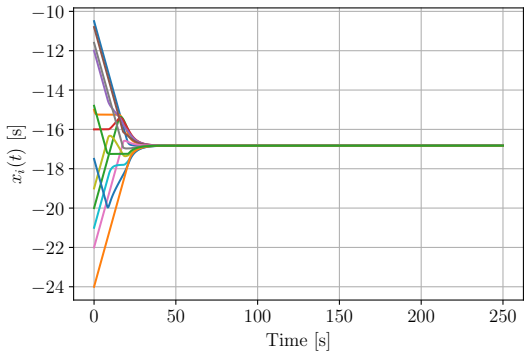

Fig. 4: State dynamics $x_i(t)$ for the ring topology

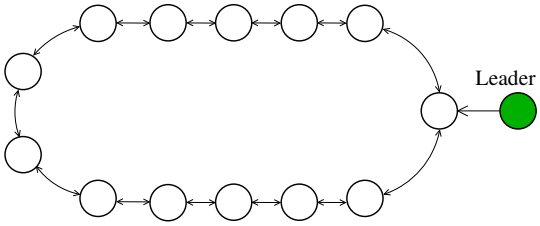

Fig. 5: Ring topology with a leader node

the synchronization time would not shrink further. Only when consensus is almost reached and the states of the agents are considerably close to each other will the controller output exit the saturation to reach zero in a shorter time. Finally, when the number of agents N is expected to be large, the synchronization time tends to be longer. For this reason, it is preferable to set higher values for the controller gains in order to reduce such transient. Simulations for different values of the k_i's is omitted since they would bear similar results but with a different transient duration before reaching consensus as just explained.

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, where each node in the ring represents a client and all the clients in the network watch the same live streaming event from different devices. In this topology, each node can communicate only with two neighbours.

Figure 4 shows that consensus is reached at roughly -17 seconds, which means that $T_i(t) = t - 17$, $\forall\, i \in \mathcal{V}$, 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 5 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 6) after about 150s. Notice that the dashed black line in the figure represents the state set by the leader node. Moreover, the transient could be made smaller if the leader is

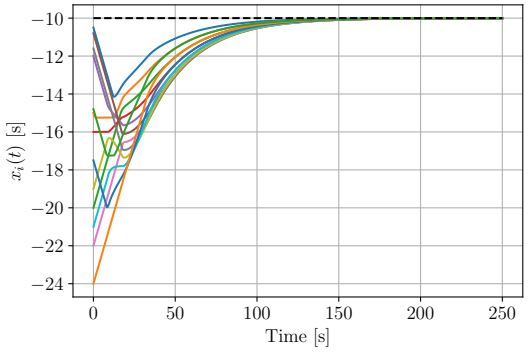

Fig. 6: State dynamics $x_i(t)$ for the ring topology with a leader node

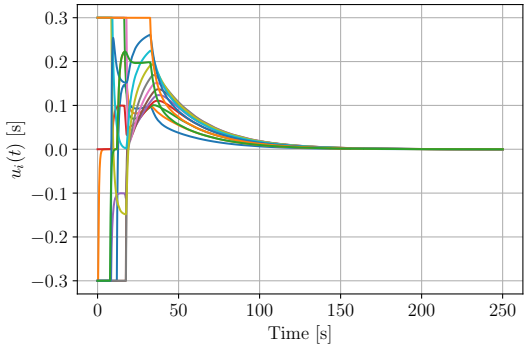

Fig. 7: Control inputs $u_i(t)$ in the case of a ring topology with a leader node

allowed to communicate also with other nodes or by properly increasing the gains k_i. Once consensus is achieved, it will result that $T_i(t) \simeq T_j(t) \simeq t - 10$, $\forall\, i, j = 1, \ldots, N$, which implies that all the clients are synchronized with a delay lower than the leaderless case (Figure 4).

Figure 7 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. The figure shows that for some clients it is necessary to increase the playback rate to make it greater than one because their initial delay was higher. On the other hand, for those clients with a low initial delay, the playback rate is made less than one at the beginning to 'meet' the others.

Let us now consider a different topology of the network composed of three groups of clients. Each client in a group is connected to the others through several edges, while groups are connected with fewer edges as depicted in Figure 8. Notice that this graph, as the previous cases, contains 13 nodes with the same initial states and control input bounds. We consider a fourteenth leader node (green node in the

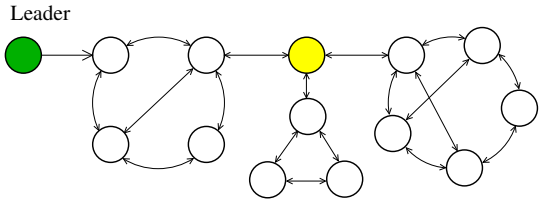

Fig. 8: Network topology with groups of clients and a leader

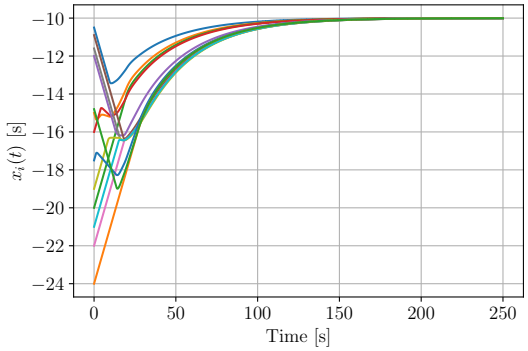

Fig. 9: State dynamics $x_i(t)$ for the topology of Fig. 8

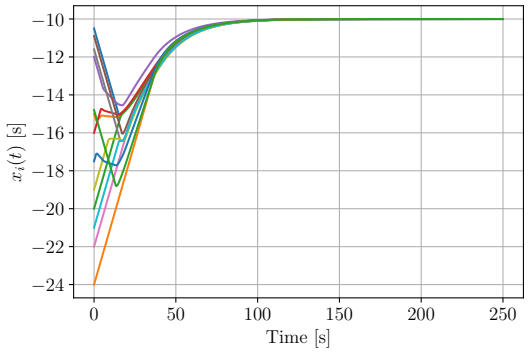

Fig. 10: State dynamics $x_i(t)$ with the leader marked by the yellow node in Fig. 8

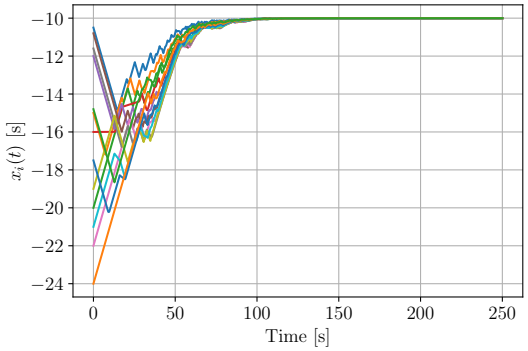

Fig. 11: State dynamics $x_i(t)$ for the ring topology in the event-triggered case

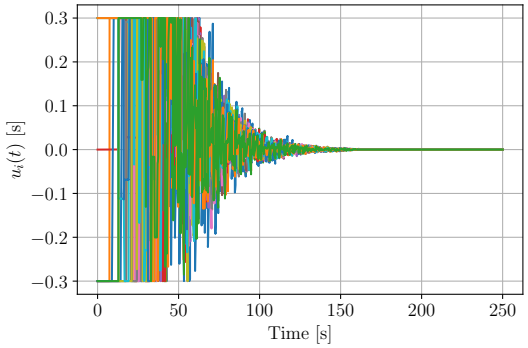

Fig. 12: Control inputs $u_i(t)$ for the ring topology in the event-triggered case

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 about 150 seconds (Figure 9) due to the different network topology and the specific node influenced by the leader. If we suppose the leader communicates with the yellow node instead, the time required to obtain consensus drops to 100 seconds (Figure 10).

To avoid unnecessary exchange of information through a continuous communication among users, we consider an event-triggered control. We set $\gamma = 10^{-4}$, $\alpha_i = 10$ and $\beta_i = 0.1$ for each $i \in \mathcal{V}$. Figure 11 shows the trend of the states in the case of leader-following approach for the ring topology. As it can be observed, convergence is still guaranteed in approximately the same time as the continuous communication case. Notice that the evolution of the states is less smooth due to the abrupt changes in the control actions (Figure 12), which, as desired, converge to zero. More precisely, once the states get close enough to each other and to the leader so that the difference of the neighbouring states is less than γ, the control inputs, already close to zero, are set to zero. This way, the states remain constant, practically synchronized, and will no longer exchange information. In Figure 13, it is possible to see the triggering time instants in which an agent updates its control action and sends its new state to the neighbours. Over a total simulation time of 500 seconds, the agents exchange information only for the first 280 seconds (roughly), after which no additional communication is needed. Another important point is that users triggered an average of only 87 events. This translates into a considerable decrease in information exchanged among users, thus implying the effectiveness of the distributed event-triggered control approach.

Finally, the same considerations could be made for other topologies, such as the one depicted in Figure 8, of which we report only the triggering time instants in Figure 14. Also in this case, a continuous exchange of information is avoided and only a limited number of events is triggered. Notice that, whilst the simulation time is 500 seconds, the figure shows only the first 200 seconds since nothing happens afterwords.

Basically, there are some nodes, depending on the network topology, that have more influence than others. Since the topology is not usually known a priori, the more nodes the leader is able to influence, the faster the achievement of synchronization. Moreover, convergence is guaranteed independently of the value of the initial states, the number of nodes

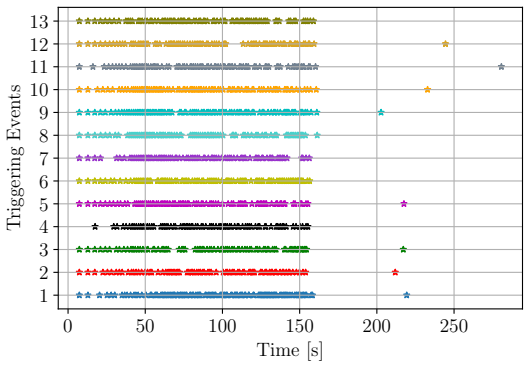

Fig. 13: Triggering times for each agent in the ring topology

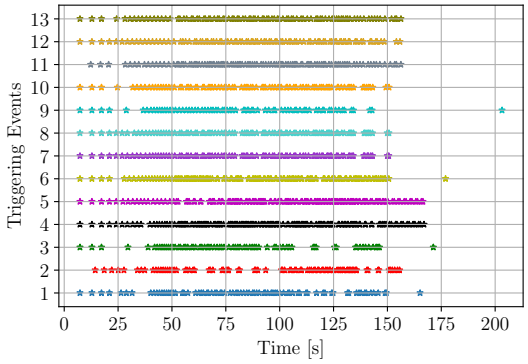

Fig. 14: Triggering times for each agent in the topology of Fig. 8

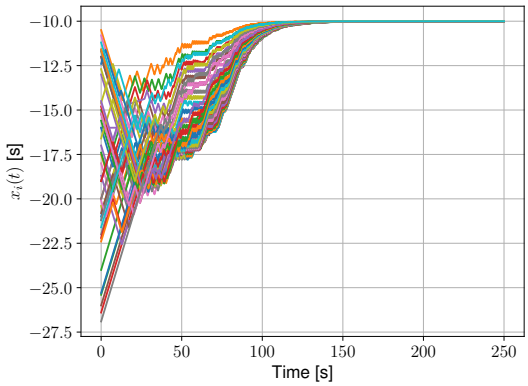

Fig. 15: State dynamics $x_i(t)$ for the ring topology in the event-triggered case with $N = 50$

involved, the number of nodes influenced by the leader and the network topology, provided the graph remains strongly connected. In fact, variations of such parameters will affect the synchronization time. In particular, when the number of users becomes large, the time needed to reach synchronization increases due to the higher amount of messages that have to be exchanged among users. The only action that can be undertaken in an attempt to reduce the transient is to increase the control gains k_i. To this end, Figure 15 shows the leader-following event-triggered control for the ring topology in the case of $N = 50$ nodes. To reduce the synchronization time, we have set $k_i = 10, \forall i \in \mathcal{V}$. It can be observed that the duration required for the agents to achieve consensus is approximately 150 seconds, slightly longer than the case of $N = 13$ agents (see Figure 11). Furthermore, it has been verified that, out of 500 seconds of simulation, the agents exchange information only for the first 280 seconds[3].

Notice that the synchronization time is also influenced by the saturation value δ that, according to a recent study, should not exceed 10% of the original rate not to impact the QoE significantly [33]. Moreover, the initial delays associated with the clients are not usually high. Specifically, the number of available video segments encoded by the DASH standard when a client connects to the server is of the order of 5-10. Considering the typical case of segments of duration $\tau = 5\,\text{s}$, the clients' playback time can be delayed with respect to the video provider of an amount that does not exceed 50 seconds. On the other hand, it is not possible to achieve perfect synchronization with the provider because at least one chunk has to be encoded and stored in the circular buffer for download. As a consequence, in the best case and assuming $\tau = 5\,\text{s}$, the minimum delay is 10s (two chunks), i.e. the value imposed by the leader in our simulations.

Finally, as stated in Section I, unlike most existing solutions, the proposed approach has the advantage of being decentralized and not requiring any change in the protocols used or in the ABR algorithm. This allows scalability and no ad-hoc web-based solutions such as in [6], [7], [8]. Unlike works proposing a decentralized approach [9], [10], our methodology does not consider a fully connected graph but just a strongly connected one. Therefore, the present work shows that it is possible to achieve decentralized playback time synchronization without acting on a specific protocol and in a scalable way by leveraging consensus theory. Note that conducting a performance comparison is beyond the scope of this work and is reserved for future research.

[3]The plot of the triggering times is omitted due to space limitations

VI. 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. Moreover, we have proposed an event-triggered control approach to reduce the information exchange and remove the assumption of continuous communication among users. In this case, clients are only required to share the value of their playback time, which is available at the player. Simulations on different network topologies prove the effectiveness of our approach, which guarantees asymptotic synchronization along with the satisfaction of playback rate saturation constraints independently of the protocols adopted. Future work will focus on the experimental evaluation of the proposed approach on real live streaming scenarios. Another direction for future work is to conduct a comparison of the proposed event-triggered approach with existing event-triggered methods.

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 MMSys*, 2011, pp. 133–144.

[3] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia, "A survey on quality of experience of http adaptive streaming," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 469–492, 2014.

[4] 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.

[5] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over HTTP," in *Proc. of ACM SIGCOMM*, 2015, pp. 325–338.

[6] 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*, 2017, pp. 524–530.

[7] 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.

[8] M. O. van Deventer, H. Stokking, M. Hammond, J. Le Feuvre, and P. Cesar, "Standards for multi-stream and multi-device media synchronization," *IEEE Communications Magazine*, vol. 54, no. 3, pp. 16–21, 2016.

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

[10] M. Montagud, F. Boronat, and H. Stokking, "Design and simulation of a distributed control scheme for inter-destination media synchronization," in *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*. IEEE, 2013, pp. 937–944.

[11] G. Manfredi, L. De Cicco, and S. Mascolo, "Synchronizing live video streaming players via consensus," in *Proc. of European Control Conference*, 2021, pp. 1062–1067.

[12] M. Kalman, E. Steinbach, and B. Girod, "Adaptive media playout for low-delay video streaming over error-prone channels," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 6, pp. 841–851, 2004.

[13] S. Park and J. Kim, "An adaptive media playout for intra-media synchronization of networked-video applications," *Journal of Visual Communication and Image Representation*, vol. 19, no. 2, pp. 106–120, 2008.

[14] Y.-F. Su, Y.-H. Yang, M.-T. Lu, and H. H. Chen, "Smooth control of adaptive media playout for video streaming," *IEEE transactions on multimedia*, vol. 11, no. 7, pp. 1331–1339, 2009.

[15] 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.

[16] M. Roccetti, V. Ghini, G. Pau, P. Salomoni, and M. E. Bonfigli, "Design and experimental evaluation of an adaptive playout delay control mechanism for packetized audio for use over the internet," *Multimedia Tools and Applications*, vol. 14, no. 1, pp. 23–53, 2001.

[17] S. B. Moon, J. Kurose, and D. Towsley, "Packet audio playout delay adjustment: performance bounds and algorithms," *Multimedia systems*, vol. 6, no. 1, pp. 17–28, 1998.

[18] L. Roychoudhuri, E. Al-Shaer, and G. B. Brewster, "On the impact of loss and delay variation on internet packet audio transmission," *Computer Communications*, vol. 29, no. 10, pp. 1578–1589, 2006.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] H. K. Khalil and J. W. Grizzle, *Nonlinear systems*. Prentice hall Upper Saddle River, NJ, 2002, vol. 3.

[24] 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.

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

[26] X. Yi, T. Yang, J. Wu, and K. H. Johansson, "Distributed event-triggered control for global consensus of multi-agent systems with input saturation," *Automatica*, vol. 100, pp. 1–9, 2019.

[27] Y. Xie and Z. Lin, "Global leader-following consensus of a group of discrete-time neutrally stable linear systems by event-triggered bounded controls," *Information Sciences*, vol. 459, pp. 302–316, 2018.

[28] L. Ding, Q.-L. Han, X. Ge, and X.-M. Zhang, "An overview of recent advances in event-triggered consensus of multiagent systems," *IEEE transactions on cybernetics*, vol. 48, no. 4, pp. 1110–1123, 2017.

[29] K. Zheng, H. Fan, L. Liu, and Z. Cheng, "Triggered finite-time consensus of first-order multi-agent systems with input saturation," *IET Control Theory & Applications*, vol. 16, no. 4, pp. 464–474, 2022.

[30] H. Zhang, D. Yue, X. Yin, S. Hu, and C. xia Dou, "Finite-time distributed event-triggered consensus control for multi-agent systems," *Information Sciences*, vol. 339, pp. 132–142, 2016.

[31] Y. Dong and J.-g. Xian, "Finite-time event-triggered consensus for non-linear multi-agent networks under directed network topology," *IET Control Theory & Applications*, vol. 11, no. 15, pp. 2458–2464, 2017.

[32] J. Fu, Y. Wan, and T. Huang, "Event-triggered finite-time practical consensus of multiagent systems with general directed communication graphs," *International Journal of Robust and Nonlinear Control*, vol. 30, no. 17, pp. 7255–7277, 2020.

[33] P. Pérez, N. García, and Á. Villegas, "Subjective assessment of adaptive media playout for video streaming," in *Proc. of International Conference on Quality of Multimedia Experience (QoMEX)*, 2019, pp. 1–6.

Gioacchino Manfredi received the Control Engineering degree (Hons.) from Sapienza University, Rome, Italy, in 2018. He received the Ph.D. degree in electric and information engineering from the Polytechnic of Bari, Bari, italy, in 2023. He is currently an Assistant Professor in control system theory with the Polytechnic of Bari. His research interests include adaptive video streaming, QoE and QoS management for video streaming services, 360° video streaming, consensus in multi-agent systems, and deep reinforcement learning techniques for nonlinear control.

Vito Andrea Racanelli (Member, IEEE) received the Control Engineering degree (Hons.) from the Politecnico di Bari in 2021. He is currently a PhD student in Control Engineering at the Polytechnic of Bari. His research interests are focused on autonomous navigation systems, predictive control algorithms, sensor-fusion, robust control, and embedded control systems. He is a member of the IEEE, of the ACM and of the Robotics & Automation Society of the IEEE.

Luca De Cicco Luca De Cicco (Member, IEEE) received the Computer Science Engineering degree (Hons.) and the Ph.D. degree in Information Engineering from the Politecnico di Bari, Bari, Italy, in 2003 and 2008, respectively. He is currently an Associate Professor at the Politecnico di Bari. He has held visiting positions with the University of New Mexico, Albuquerque, NM, USA (2007); Ecole Superieure d'Electricitè, Paris, France (2012); and the Laboratory of Information, Networking and Communication Sciences, Paris (2013, 2014). He has coauthored more than 60 papers published in international journals, books, or conferences. His main interests are the modeling and design of congestion control algorithms for multimedia transport, congestion control for Web real-time communication, adaptive video streaming, 360 video streaming, QoE-aware resource allocation, robotics, reinforcement-learning. He has been the General Chair of the ACM Multimedia Systems conference in 2024.

Saverio Mascolo (Fellow, IEEE) received the Laurea degree (Hons.) in electronics engineering and the Ph.D. degree from the Politecnico di Bari, Italy, in 1991 and 1994, respectively. He is currently a full professor at the Politecnico di Bari, where he has been the Head of the Department of Electrical Engineering and Information Science (2015-2021). He was a Postdoctoral Researcher (1995) and a Visiting Researcher (1999) with the University of California at Los Angeles, Los Angeles, and a Visiting Consultant (2002-2004) with the University of Uppsala, Sweden. He has authored or coauthored more than 140 papers in international journals, books, or conferences. He has worked on intelligent manufacturing systems, deadlock avoidance, nonlinear control, chaotic systems, crypto communications using observers, Internet congestion control, adaptive video streaming, quality of experience in multimedia delivery, and congestion control for Web real-time communications. He was recipient of the Cisco Research Award in 2013 and the Google Faculty Award in 2014. He has been an Associate Editor of the IEEE Transactions on Automatic Control and of the IEEE/ACM Transactions on Networking. He is currently Associate Editor of Computer Networks (Elsevier).