Control-driven Media

A unifying model for consistent, cross-platform multimedia experiences

Ingar M. Arntzen*, Njål T. Borch†, Anders Andersen‡
*NORCE Norwegian Research Centre, Tromsø, Norway
Email: inar@norceresearch.no
†Schibsted, Tromsø, Norway
Email: njal.borch@schibsted.com
‡UiT The Arctic University of Norway, Tromsø, Norway
Email: anders.andersen@uit.no

***Abstract*—Targeting a diverse consumer base, many media providers offer complementary products on different platforms. Online sports coverage for instance, may include professionally produced audio and video channels, as well as Web pages and native apps offering live statistics, maps, data visualizations, social commentary and more. Many consumers are also engaging in parallel usage, setting up streaming products and interactive interfaces on available screens, laptops and handheld devices. This ability to combine products holds great promise, yet, with no coordination, cross-platform user experiences often appear inconsistent and disconnected.**

We present *Control-driven Media (CdM)*, a new media model adding support for coordination and consistency across interfaces, devices, products and platforms, while also remaining compatible with existing services, technologies and workflows. CdM promotes online media control as an independent resource type in multimedia systems. With control as a driving force, CdM offers a highly flexible model, opening up for further innovations in automation, personalization, multi-device support, collaboration and time-driven visualization. Furthermore, CdM bridges the gap between continuous media and Web/native apps, allowing the combined powers of these platforms to be seamlessly exploited as parts of a single, consistent user experience.

CdM is supported by extensive research in time-dependent, multi-device, data-driven media experiences. In particular, State Trajectory, a unifying concept for online, timeline-consistent media control, has recently been proposed as a generic solution for media control in CdM. This paper makes the case for CdM, bringing a significant potential to the attention of research and industry.

***Keywords*—multi-platform, media control, continuous media, data-driven media, interactive media, orchestrated media**

I. Introduction

The landscape of media production and *over-the-top (OTT)* delivery holds immense potential. Advanced production tools, automated AI-based technologies, and a wealth of data sources come together, often in multi-step, distributed production chains. On the client-side, highly capable consumer devices offer further opportunities for adaptation, customization, interactivity and data-driven graphics. Concurrently, consumer preferences are evolving: ranging from those who favor the traditional one-size-fits-all broadcast experience, to those seeking multi-device immersion, interactive engagement, customized multi-device setups, personalized narratives, social interactivity, or accessibility features. This diversification poses a significant challenge for media providers aiming to offer advanced, high-quality user experiences to a sizable audience, while also managing costs and complexity.

To address needs for richer and more varied experiences, many media providers are offering alternative products on different platforms. For instance, coverage of major sport events may include live produced video channels, VoD services, as well as Web or Native apps with support for live feeds, interactive data visualization, and social integration. Many viewers also see these as complimentary offerings and engage in combined usage as a way of further enriching their experiences. This though, may be less rewarding than perhaps anticipated. Each product must be configured separately, and there is typically no coordination across platforms. Differences in production delays may also lead to inconsistencies, confusion and spoilers. This significantly limits the value of combined usage, possibly even driving viewers back to traditional single product coverage.

We envision a new class of cross-platform media experiences, where independent products and interfaces may be flexibly combined and coordinated to form a single, consistent, user experience. Users can then enjoy a spectrum of experiences, from lean-back entertainment to lean-forward engagement, from single device to multi-platform immersion, or from private to social experiences. Furthermore, by opening up for cross-platform usage, media providers can provide advanced and uniquely adapted user experiences in a cost effective way, by leveraging the combined powers of existing infrastructure and services, while also ensuring quality and brand control for the user experience as a whole.

Unfortunately, issues with cross-platform user experiences are not easily addressed within the current paradigm. Different products are built from independent technology stacks and define entirely separate user experiences. While some solutions exist for coordination and consistency, they are often application-specific or limited to specific technology options, such as data formats, distribution protocols or presentation frameworks. In contrast, we argue that cross-platform coordination must be independent from platform-specific solutions, and that support for cross-platform media experiences should

instead be addressed as a fundamental feature of the media model.

To this end, we propose *Control-driven Media (CdM)*, a new media model with built-in support for consistent, cross-platform user experiences. CdM promotes control as a principal resource type in media systems. Through online control sharing, applications can orchestrate connected interfaces and render data sources and media content consistently across platforms. Moreover, as CdM is compatible with existing infrastructure and workflows, adoption can be incremental and cost-efficient. To the best of our knowledge, this is the first attempt to provide a general solution for consistent, cross-platform user experiences.

The goal of this paper is to present CdM and the considerable opportunities associated with the model. This paper is structured as follows. Section II presents current solutions for cross-platform media experiences, and challenges leading up the problem statement in Section III. Section IV outlines the current media model, before Control-driven Media is introduced in Section V. Section VI discusses key opportunities for online sport coverage and indicates how CdM may support such developments. Section VII covers key technical challenges, whereas Section VIII includes evaluation and references to implementation and supporting research. Section IX provides a brief discussion before the paper is concluded in Section X.

II. Background

The online world provides unprecedented opportunities for online media, with a wealth of tools and platforms for capturing, editing, distributing, and rendering of data and media content. Moreover, the ongoing AI revolution promises further opportunities for automation, content generation, personalization, and more. To fully exploit this potential, media systems must meet an increasingly complex set of demands. Media providers seek to build their brand through high-quality user experiences and exciting narratives. There might also be a race for advanced features, such as impressive graphics, interactive engagement, device adaptation, social integration, and/or multi-device support. This, though, must be balanced with technical and financial considerations, including performance, scalability, complexity and costs. Users, on the other hand, seek media experiences which are relevant, exciting and distraction free. Beyond this, users are diverse, and preferences will likely diversify further as offerings become more advanced. For example, some users might prefer lean-back storytelling, while others want to engage and actively shape the experience for themselves, or for others. Some prefer generic broadcast coverage, while others would like a more personalized narrative. Some want experiences to be social, some seek multi-device immersion, and some have specific accessibility requirements.

Supporting diversity is hard though, and certain demands may even appear conflicting in terms of technology options. In particular, we recognize two technical approaches for audiovisual user experiences, *continuous media* and *data-driven media*. Media production based on continuous media types (e.g. audio and video) supports high-quality storytelling through precise mixing and scheduling of video sources, audio tracks, and graphical elements. This approach can provide user friendly, lean-back, and action packed user experiences, yet primarily the same experience to all viewers. In contrast, data-driven media (e.g. Web/native applications) offer dynamic experiences with flexible options for interactivity, visualization, adaptation, personalization and collaboration. However, data-driven media provide limited support for weaving complex time-dependent, lean-back narratives, particularly if they involve multiple data sources and/or rendering technologies.

To address complex demands from users and media providers, it appears necessary to leverage the combined power of these approaches. This section presents common ways of combining technologies, highlighting strengths and limitations with each approach.

Embedded Media: A common way to combine technologies is to embed continuous media within a data-driven interface. On the Web platform, for example, audio and video content may be embedded in Web pages using the *HTML5 Media element* [1]. This way, soccer coverage may include both a video stream and a feed of match events. However, despite being part of the same layout, embedded components technically define separate user experiences. For instance, the video pause button might not apply to the feed of match events, which will continue to report game developments. This creates an inconsistency, where the two components display data from the same event, but from different time frames. Media providers may address this by introducing additional coordination between components. For instance, on the Web platform, media players and feeds may be controlled from code. However, this leads to custom solutions for different applications and platforms, and complexity increases with the number of coordinated components.

Overlays: A related method is to layer a transparent, data-driven interface on top of a video display. This technique can for instance be used in video production, to burn graphics onto a video stream. Alternatively, graphics can be overlaid in the user interface, using a *z-index* or similar layering concepts supported by the layout system. This provides additional options for individualized graphics and interactivity, but requires that overlay graphics are synchronized with video progression. This can be accomplished by defining cues, hooks, or events on the video timeline. In the Web platform, this is supported by the concept of data tracks [1] integrated with the HTML5 media element. While this provides coordination between continuous and data-driven media, the approach is most useful when the experience is limited to a single video asset within a single interface or layout.

HbbTV: HbbTV (Hybrid Broadcast Broadband TV) [2] is a standards initiative and platform allowing HTML-based graphics to be overlaid on broadcast video content. In HbbTV, the focus is on the set-top box or SmartTV, which supports IP-based data access as well as traditional broadcast distribution. By exposing the media clock of the broadcast stream to the HTML5 processing environment, program specific or even personalized overlays may be presented on the TV display. Broadcasters may define and deploy such overlays as HTML

applications associated with a channel or program. Furthermore, the concept of overlays is also extended to companion devices. This means that smart phones may connect to the HbbTV device to access additional contents or interactive capabilities, synchronized with the broadcast clock [3]. However, adoption of HbbTV has been slow, as it requires development and deployment of HbbTV enabled consumer devices. Moreover, as the approach hinges on the existence of a physical device, it does not easily generalize to other media domains or beyond the home environment.

Low latency streaming: Another way to combine media content and data-driven graphics, is to coordinate delivery of multiple data streams. For example, an online lottery might want to stream a video from the lottery drawing, and also provide a separate data stream with the winning numbers, for visualization. This requires some coordination between streams, or else the graphics might spoil the suspense of the video, or even make the lottery appear suspect. One approach is to minimize latency in streaming, thereby also limiting the potential misalignment of streams. Global CDN providers such as Akamai [4], Cloudflare [5] and Fastly [6] provide scalable, low-latency streaming solutions based on Dash [7], HLS [8] or WebSocket [9] protocols. Another option is to leverage support for synchronized stream delivery or fixed end-to-end delays. For instance, in IP-based networks, fixed end-to-end delay may be achieved by transmitting data to a client ahead of time, and then scheduling the delivery of data to the application in reference to a media clock. However, such methods have some limitations. Ultra-low-latency streaming may be expensive and also provide reduced user experience due to network jitter. Moreover, synchronizing content using the distribution system increases the complexity of the distribution system itself, which goes against current practices of stateless servers to limit cost and increase scalability.

Object-based Media: In Object-based Media (ObM) [10]–[12], the challenge of combining technologies is addressed already in production. ObM targets increased support for adaptation, personalization and interactivity in broadcasted experiences. The key idea is to represent media as objects, and let client devices be responsible for assembly into rendered presentations. This way, clients may assemble media experiences differently, sensitive to device capabilities, local context or user preferences, and leverage data-driven technologies for visualization and interactivity. However, the approach requires changes to media formats, with implications for existing workflows and tools. This is speculated to be a key reason for slow adoption [13]. In addition, ObM is still bound to the single stream broadcast metaphor. This means that objects are packaged and distributed as one asset, even if each viewer will only make use of a subset of objects. Moreover, the approach does not provide any particular support for multi-device media consumption.

Leave it to the user: The last approach, and perhaps the standard solution, is to leave the combination of technologies as an exercise for the viewer. *Formula 1 (F1)* online coverage provides an example of this. *F1TV* [14] offers a number of video streams to choose from, including the professionally produced World Feed, onboard vehicle cameras with team radio for that driver, a produced pit lane feed and even a channel with lap times in tabular form. In addition, the *F1 App* [15] provides an interactive race map and numerous data visualizations for detailed race statistics, which also supports manual time shifting. Viewers may then select the most relevant video feeds and visualizations, and also use multiple devices to follow multiple interfaces in parallel. As such, this approach supports high levels of customization and personalization. At the same time, to realize this potential, viewers must do all the work. For example, synchronization between interfaces is very important in a fast-paced sport like F1, yet cumbersome to achieve manually. Moreover, users can not easily know which camera feed is most relevant at any time, or when to switch between views. In short, this approach encourages viewers to act as producers, and leaves media providers with little control over the quality of user experiences.

A. Challenges

It appears that combinations of continuous media and data-driven applications have been attempted in different parts of the technology stack, in production, in distribution and in presentation, yet each method comes with limitations. Still, combining technology platforms remains an attractive prospect. Online F1 coverage, for instance, showcases a significant potential if only some coordination could be provided between products.

In current systems though, there is limited support for such coordination. Figure 1 *(left)* illustrates 4 product interfaces (green rectangles), each defining its own user experience (bubble). This leaves the viewer to mediate between separate user experiences, either compensating mentally for any inconsistencies between them, or manually attempting to correct them, for instance by time-shifting or configuring individual interfaces. Neither option is particularly appealing.

Figure 1 *(right)* illustrates the basic idea of cross-platform media experiences. Here separate product interfaces (green rectangles) contribute to a single, consistent user experience (bubble). Consistency implies (i) that interfaces operate in reference to a shared media clock, and (ii) that user interactivity is not limited to one product, but applies to the experience as a whole. For instance, if the user selects a different F1 driver, this might potentially affect all interfaces in different ways, including overlayed video graphics in F1TV and the race map visualized by the F1 App.

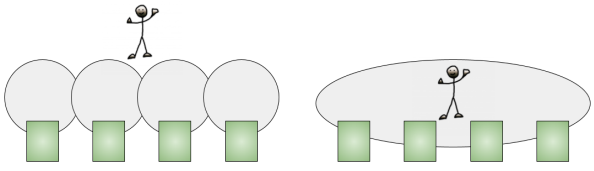

Fig. 1. *(Left)* User engaging with 4 distinct user experiences (bubbles), defined by 4 independent products (green rectangles). *(Right)* The same 4 products are instead contributing to a single, consistent user experience (user inside bubble).

To support the notion of cross-platform media experiences, we argue that a solution to coordination is needed, which is not

limited to any particular application or platform. We envision a new media model with built-in support for coordination. In this model, media products can still be developed for single platform usage, yet optionally be exploited as parts of larger, consistent, cross-platform media experiences.

III. Problem Statement

Create a media model with built-in support for coordination between independent devices, technology types, platforms and product interfaces. The new model shall

1) support consistent user experiences across platforms *(consistency)*
2) support the combination of continuous and data-driven media *(bridge)*
3) support integration with existing media systems *(compatible)*
4) support high flexibility while limiting complexity *(practical)*
5) support key trends such as automation, adaptation, personalization *(future-proof)*

IV. Current Media Model

Even though *continuous media* and *data-driven applications* represent different technical approaches, we regard them as instances of the same media model. Figure 2 illustrates this model as a 3-step processing chain (rectangles). The data flow is left-to-right, from data (black) on the left, through assembly (blue) and rendering (green), before reaching the user experience on the right (bubble). User-interactivity (not illustrated) flows in the opposite direction, triggering data updates (black arrow) or control actions (red arrow). Processing steps (data, assembly, render) may be executed within the same process, or be separated by a communication link or a network connection.

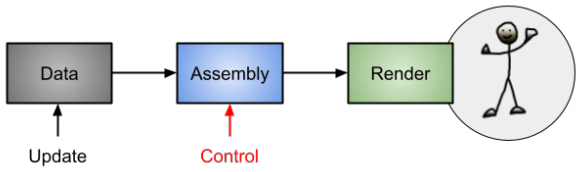

Fig. 2. Conceptual sketch for current media model. The data flow is left-to-right, from data (black) on the left, through assembly (blue) and rendering (green), into the user experience on the right (bubble).

- **Data.** Data represents resources for media experiences. Resource types may include common media formats such as audio, video, images, XML/JSON-data, declarative layout or stylesheets. Resource abstractions may include files, streams, databases, or datasets. Resources may be static or dynamic, local or accessed via a network. Access may also be restricted through concepts of ownership and access credentials.
- **Assembly.** Assembly (blue) implements a controlled conversion from data sources to render state. This conversion may include processing steps such as selection, filtering, merging and transformation of data from multiple sources. Moreover, assembly is open to dynamic control (red arrow), defining for instance which data sources are selected for rendering, how data is converted into render state, and when. Assembly may also be associated with a media clock and expected to update render state consistently with respect to time progression. This is commonly referred to as playback or sequencing.
- **Render.** Render (green) is a function converting render state into visual, auditory or other real-world effects. Render components are assumed to be software components, acting as low-latency or fixed-latency proxies for locally connected output devices, such as displays and loudspeakers.
- **User Experience.** The user experience (bubble) is defined by rendered effects (i.e., audio, visual, or other). The user may also interact with the media experience, using local input devices such as keyboards, pointing devices and more. User interactivity may result in updates to data sources (black arrow) or control actions (red arrow) targeting the assembly process.

In this model, the assembly step represents the beating heart of the media experience. As either data or control inputs change, the assembly process must continuously reevaluate and adjust the output render state. Over time, this produces a time-sequence of render states which defines the progression of the media experience.

A. Continuous Media

Figure 3 illustrates *over-the-top (OTT)* audio and video production as an instance of the current media model. In this context, data access and media assembly primarily occurs within a production environment. For instance, video production may involve a large set of data sources and media contents, mixed together into a single audio or video asset and streamed over a network for playback on viewer devices. Provider-side media assembly may also involve a large production team controlling a number of production parameters, including camera positioning, lighting, sound processing, graphics and visualization, as well as mixing and scheduling. A few aspects of assembly are open to local control by viewers. For instance, media players typically allow viewers to adjust the volume, pause/resume, and select subtitle tracks.

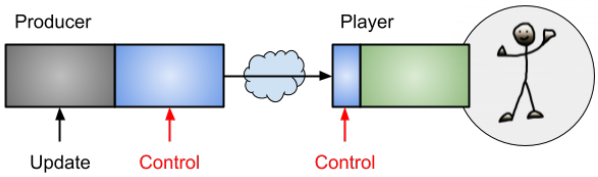

Fig. 3. Data assembled into video or audio streams at the provider-side, then distributed for client-side rendering by a media player.

B. Data-driven Applications

Figure 4 illustrates data-driven Web or native applications as instances of the current media model. In this context,

data sources are hosted by servers. Clients connect to fetch or stream data over the network, or receive pushed data. Assembly and rendering are handled at the client-side. For instance, in Web interfaces, clients convert data sources and application state into render state managed by the *Document Object Model (DOM)* [16]. The conversion is defined in application code and may include selection or filtering of data, and also transformations or combinations of data. Moreover, interactive control by the user may trigger changes to local control state or network accessible data sources.

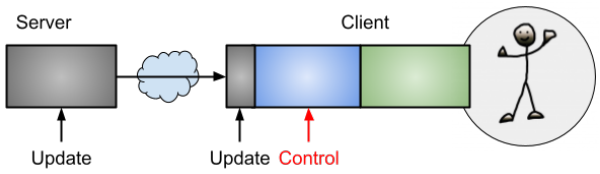

Fig. 4. Web and native applications perform assembly and rendering at the client-side, based on data fetched or streamed from online servers.

V. Control-driven Media

We envision cross-platform coordination as an inherent capability of the media model. Recognizing the central role of control in media (see Section IV), we propose a revised media model, where control is redefined from local signal to an independent, stateful resource, and extended with built-in support for cross-platform coordination. This transition, we argue, has profound implications for media applications and user experiences, and alters the status of control as an object of research, from application-specific interface feature to principal system component.

Figure 5 illustrates the revised media model. Similar to Figure 2, the model describes a 3 step technology stack (Data, Assembly, Render). The difference though, is that control (red rectangle) is regarded as a special kind of data source, and included in the first step (Data). Like data sources, control sources may be hosted by dedicated online services and accessed by connecting clients.

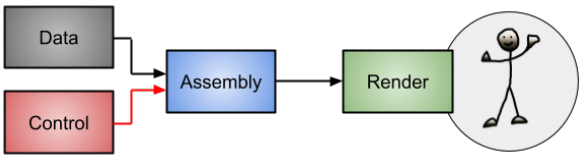

Fig. 5. Control-driven Media, with control represented as a special kind of data source.

Importantly, the revised model does not represent a radical change. Data sources (black) and render components (green) are not affected. Assembly (blue) will react to state changes in control sources, instead of live control inputs, but otherwise remain the same. Also, user interactivity (not illustrated) will not target Assembly (blue) directly, but indirectly through modification of control sources (red). Beyond this, the model remains the same.

This model is referred to as *Control-driven Media (CdM)*. The name highlights how media experiences can be directed through manipulation of control sources. For example, control sources may define which resources and rendering components are active (e.g. video sources, audio tracks, datasets, layout templates, graphical elements). Control sources may also define values for a variety of interface parameters (e.g. style properties, playback offsets, audio levels, viewport positions). By manipulating the value of such control sources, media experiences will change in predictable ways. As such, control sources represent aspects of application behavior or appearance which are explicitly opened up for external control. The following are defining properties of Control-driven Media.

1) Control is defined as a stateful, shareable resource, local or network accessible.
2) Control is defined in reference to a timeline.
3) Control defines the experience.
4) Assembly is an independent step.

1) Control is a stateful, shareable resource, local or network accessible: Control sources may be local objects. However, with control represented as online resources, considerable opportunities arise with respect to sharing and exchange of control. In particular, control may be distributed in real-time between any connected clients across various interfaces and platforms. This may allow media providers to implement production control across a distributed production chain and also across consumer devices. Effectively, this makes consumer devices an integral part of the production infrastructure, blurring the distinction between production and presentation. Online control may also provide a valuable integration point for automated processes, for instance allowing cloud-based AI-agents to remote control user experiences in accordance with user preferences. Moreover, online control can be shared between media providers and viewers, opening up for increased collaboration. For instance, when the viewer requests more detailed graphics, the production system can access the online control state for this viewer and implement appropriate changes through modification of relevant control sources.

Control as a stateful resource opens up for a variety of control patterns. Online controls may be private, public, or limited to groups. Access restrictions may also discriminate between roles such as owner, editor or viewer. Online control can support multiple patterns for control exchange (1:1, 1:N, M:1 or M:N) and support both one-way (asymmetric) and multi-way (symmetric) control relations.

2) Control is defined in reference to a timeline: With control as a network accessible resource, latency is no longer negligible. This is particularly problematic as control signals

are often time dependent. Control actions might refer to a specific offset on a media timeline and/or describe time-dependent transitions. When transferring control signals over a network, such temporal relations must be preserved. Control may also be shared between processes operating in different time frames. For instance, on-demand consumption requires time-shifted replay of previously recorded control sequences. Moreover, in scenarios with real-time sharing, small skews may be introduced to mask network jitter and avoid buffering issues. To support this, we assert that control must support timeline-consistent sharing between processes. Timeline-consistency implies that a control signal can be captured and serialized in reference to a media clock, and reproduced correctly according to a different media clock.

> Consistency with a timeline is also a defining characteristic of continuous media. This means that control sources may be regarded as media objects in their own right, and also be described using terminology traditionally reserved for continuous media types. So, like video, control may be captured, recorded, distributed, time-shifted, rewinded and played back.

3) Control defines the experience: While CdM changes the nature of control, it maintains established distinctions between data and control. Data sources tend to be raw material for an experience, whereas control sources define a particular realization of such inputs. There might also be a certain asymmetry between the two entities, where data sources may represent large and stable datasets, whereas controls are more lightweight and dynamic. As such, control provides a basis for variation, where different experiences can be produced, without changing the data. By opening up for production and distribution of control as a separate and independent resource type, this pattern can be exploited directly in consumer interfaces, providing personalized user experiences as unique permutations of shared data sources and personalized control sources.

> CdM implies a state-based, reactive approach to control. Developers define what is considered control state in a particular application, how control state can be modified, what sharing scope and access restrictions are appropriate, and how control is coupled with application logic (i.e. assembly and interface components). Interface components typically initiate control actions, whereas assembly processes react to changes in control state. If control sources are shared online, this decoupling between producers and consumers of control may apply in the global scope.

4) Assembly is an independent step: In the current media model (see Figure 2), local control signals dictate that assembly is co-located with the controller, be it the media provider in a studio, or the user in an interactive interface. With respect to architecture, this forces a choice between provider-side and client-side assembly, with no clear middle ground. With control as an independent resource, assembly instead becomes an independent processing step, with no particular restrictions concerning location. This means that the assembly function can be shifted between locations without modification, for instance between a physical studio and consumer devices, even if rendering technologies may be different for these locations. Moreover, assembly can be split into logical steps and assigned to different nodes in a production chain, such as cloud-hosted production services, edge-nodes, and consumer devices (see Figure 6). Even though nodes may operate in different time frames, and are not necessarily directed by the same control parameters, the entire chain can still be controlled through the same mechanism.

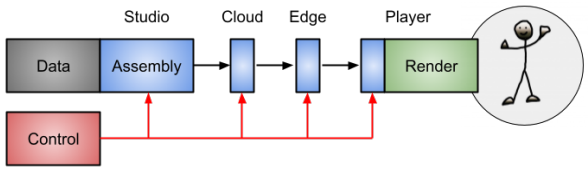

Fig. 6. A stepwise production chain, from physical production studio to consumer device, through cloud-based production and CDN/edge services. Control is time-shifted for each step to allow time for data transfer.

A. Cross-platform Media Experiences

Control-driven Media (CdM) provides a new and highly flexible foundation for cross-platform media experiences. Figure 7 illustrates three media products $\{P1, P2, P3\}$ hosted by different platforms. Collectively, these products are supported by assembly functions $\{A1, A2, A3\}$ (blue), control service $\{C\}$ (red) and data services $\{D1, D2, D3, D4\}$ (black). Dependencies for each product may be deduced by traversing the graph recursively, starting from rendering components. For instance, product $\{P2\}$ is defined by render components $\{R2, R3\}$ dependending on $\{A2, A3, C, D2, D3, D4\}$.

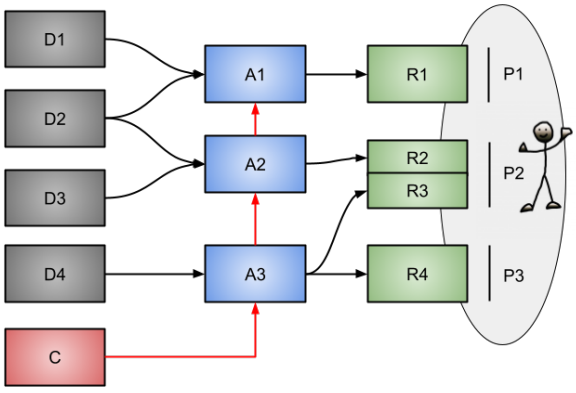

Fig. 7. Cross-platform media experience with three separate products supported by independent technology stacks.

While products $\{P1, P2, P3\}$ may be developed and used independently, they can also be combined to form a larger

cross-platform media experience. In principle, this comes down to the scope of control resources, i.e. whether control resources are defined to be exclusive for each product, or shared across multiple products. Moreover, with a light coupling between control resources and application components, aspects of control sharing may easily be changed between application, interfaces or even dynamically during a session. Importantly, this means that CdM can support cross-platform media experiences, without requiring architectural changes or significant changes to existing backend systems or render components.

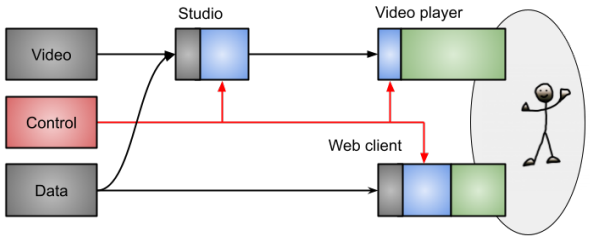

Fig. 8. Cross-platform media experience with video and data-driven visualization.

Figure 8 illustrates a combined production chain with video content and Web visualization. For instance, in the context of Formula1 racing, *Video* (top, left) could be camera feeds from the track, and *Data* (bottom, left) could be a live dataset with lap times. In the *Studio* (top, middle), a producer may then add data-driven graphics to the video content, for instance switching between a lower-third with the most recent lap-time for the featured driver, and a larger table with lap times for different drivers. The *Web client* (bottom, right) may present similar, HTML-based graphics, from the same data source. Furthermore, control sources can be shared between different steps in the production chain. This way, actions taken by the studio producer (e.g. activate or deactivate a graphics element) could also be used to direct Web clients, though time-shifted to match the timeframe of the *Video player*. The resulting user experience could then be a consistent narrative, though presented across very different technologies.

This example particularly demonstrates opportunities with synchronized secondary device presentations in broadcast, or live production of lean-back experiences with data-driven rendering technologies.

VI. Applications

Control-driven media (CdM) provides ample opportunities for innovation in media applications. This section details specific examples in the context of *Formula1 (F1)* racing in order to highlight some possibilities, though the themes discussed are likely relevant for a broad range of media applications, and also beyond the world of online sports.

A. Formula 1 Online Coverage

Over recent years, Formula 1 has strengthened its online presence, adding the F1TV [14] streaming service and the F1 App [15] to their classical broadcast offering via TV rights deals. In total, F1 caters to a worldwide audience of motor sport enthusiasts, with an estimated viewership of about 70 million per race in the 2023 season [17]. Ambitions are clearly stated at the F1 site [18]: ”Every F1 session live and on demand.” F1 coverage offers a variety of interfaces into race events, based on vast amounts of data and content streams captured by sensors, cameras and microphones – placed inside vehicles and around the racing track. Products include premixed video channels, specific camera angles or audio tracks, maps with live driver tracking, time tables, interactive visualizations of sensor data and analytics, as well as edited highlights. F1 fans enrich their own experiences by switching between different interfaces, or using multiple interfaces in parallel. In particular, by opening multiple instances of F1TV (5 simultaneous devices are allowed per account) and F1 App on different devices, more screen estate is available for a more immersive experience.

Fig. 9. Formula1 online offerings. Image credits formula1.com

B. Case: Adaptive Graphics

Graphics are an essential part of the F1 experiences, amplifying and illustrating important trends and sudden developments from an otherwise overwhelming data corpus (estimated to over a TB of data per race, just from the cars [19]). With adaptive graphics even more opportunities arise. For example, graphics could adapt to custom aspect ratio and personal preferences (e.g. ”novice”, ”regular”, ”engineer”). This is not feasible with traditional video production.

In CdM, graphics can be rendered on the client-side, as video overlays or standalone, data-driven components, while at the same time being controlled from a centralized production system. Client-side graphics can be sensitive to user interactivity and local context (e.g. available display area, gpu support, power level). Control-driven graphics may also be precisely aligned with video content and connected to live data sources. For instance, during strategically important periods, specific graphics could monitor relevant details for a specific strategic battle. Such graphics already exist, but are only shown sporadically in the World Feed, and normally only for top-team cars involved in a close battle. Furthermore, time-shifted graphics may display up to date information, and also remain open for interactive exploration.

C. Case: Immersive Multi-device

F1 is a data intensive sport, and many F1 enthusiasts are immersing themselves with different views across multiple

devices. A successful multi-device setup provides more screen estate to the user experience, and also allows capabilities of different device types to be exploited, for instance by placing video contents on large screens, and interactive features on handheld devices with touch displays. In current offerings, all configuration, synchronization and selections must be done manually by the user.

In CdM, multi-device usage could be simplified by automated configuration. For instance, in response to a pit stop period, video sources and F1 App visualizations could be switched to present relevant data, and personalized HTML-based overlay graphics could be presented consistently with both video content and app visualizations. Control sources may also be used for real-time interactivity between interfaces. For instance, driver selection in the F1 App could affect graphics and content selection across multiple devices. Moreover, control state could also define a mapping between interface components and devices, opening up for automated reconfiguration as devices join or leave the experience [20]. This could for instance be controlled by AI-based agents trained on manual configuration patterns or predefined policies.

D. Case: Lean-back, lean-forward

F1 advertises a highly customizable product, where viewers can shape their own experience by switching between F1TV streams and interacting with F1 App visualizations. While such interactive engagement is clearly attractive, it may also be taxing in the long run, and even the most forward-leaning F1 enthusiasts may want to lean-back at times, for instance during highly exciting race sequences. In current solutions though, this typically implies a set back to generic broadcast coverage, e.g. the World Feed.

With CdM, the distinction between lean-back and lean-forward coverage can be softened. For instance, by actively controlling aspects of data-driven interfaces, media providers may exploit traditionally lean-forward technologies as part of a produced narrative. Shifting between lean-back and lean-forward modes can also occur seamlessly within the user experience. For instance, switching from lean-back coverage to interactive exploration may simply be a matter of switching control sources, from official provider controls to private controls.

E. Case: Personalization

F1 caters to a highly diverse audience, and many users might appreciate specialized coverage based on preferences such as team affiliation, nationality, language, proficiency level (novice, regular, enthusiast), or accessibility requirements. Individual adaptations, though, are costly in the context of video production.

In CdM, user experiences may to a larger extent be assembled from smaller building blocks (i.e. control and data sources). This provides multiple opportunities for variation. Interfaces may be set up with different versions of data and control sources, and assembly logic may combine them in different ways. This way, CdM may provide the appearance of unique adaptation, even from a modest number control parameters. Personal preferences may also be reconfigured with immediate effects. For instance, changing proficiency level from "regular" to "enthusiast" may trigger a cascade of changes with low-level control resources, effectively transforming the experience.

F. Case: Social and 3'rd party integration

By opening up for collaboration and 3'rd party contributions, media experiences could become both richer and more social. For instance, graphics could indicate the presence of friends watching the same event, and allow viewers to join sessions, or receive reactions from friends who watched earlier. A group of friends could choose to watch the latest F1 race together online, taking turns bringing up interesting stats in the F1 App, for all to see and discuss. This notion of collaborative production could also extend to 3'rd party services providers. For instance, a national group for Ferrari fans could provide its members with exclusive contents through the official F1 platform. With a closed production system though, such integrations are often complex and also problematic with respect to content ownership.

In CdM, social and 3'rd party integration can be addressed through control sharing. For instance, by sharing control sources between friends, race events can be presented in synchrony, ensuring that all react to race events at the same time. It would also be possible to follow the experience of a friend, or actively direct some aspect of presentation - for a group. Moreover, 3'rd party content sources could be integrated directly in client facing interfaces, thereby avoiding integration with backend systems.

G. Case: Value-added time-shifted coverage

F1 is a live sport, and most F1 viewers prefer to follow live coverage. Still, F1 races are hosted at venues around the world and across many different time zones, making time-shifted consumption a more convenient option in some cases. Time-shifted coverage could also provide other benefits. For instance, production errors could be corrected, and results from time-consuming analysis could be added. A lot of information is also held back during the race itself for strategic reasons. Currently though, time-shifted offerings are limited to replay of the original video content and short edited highlights.

In CdM, time-shifted coverage is not limited to video, but may also include replays of data-driven interfaces. The latter implies that time-shifted coverage can remain interactive and open to adaptation. Moreover, control sources can be modified at any time after production, for the benefit of time-shifted viewers. For instance, post-race analysis could optimize control state for the selection and scheduling of video sources (i.e. camera angles) to perfectly capture passes and race incidents as hindsight gives perfect information. This may also apply to live viewers, as live experiences are also time-shifted to some degree, for instance due to delays in production chains for video content.

VII. Challenges

Control-driven media introduces some new technical challenges. A key challenge concerns the redefinition of *control*, from local signal to online resource. Another important challenge concerns increased complexity in *assembly*, where potentially a large number of data and control sources need to be consistently converted into render state.

A. Control

In control-driven media, control is a stateful, network accessible resource, defined in reference to a timeline.

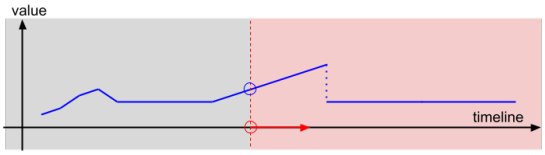

Fig. 10. A control parameter defined in reference to a timeline. The control value (blue) increases and decreases gradually, remains static for a while, increases again, before it is abruptly set to a lower value. By time-shifting the media clock (red ring), past values of the control parameter may be played back again. Illustration adapted from [21]

Challenge 1 - Generic control: The ability to share control is a fundamental property of control-driven media. Moreover, control sources must support a variety of control patterns, be shared across different time frames and across a variety of products and application types. To allow such flexible usage, control-driven media will benefit from a common abstraction for control. At the same time though, applications are different and developers must also be able to specify unique control relations and custom control logic. The challenge then is to provide a generic and flexible mechanism for control sharing, to be used across very different applications, while at the same time opening up for application-specific extensions or adaptations.

Challenge 2 - State representation: In order to be shared across a network, a serialized format must be defined for control state. One challenge is to define a representation which supports various types of control, including discrete and dynamic state changes. Another challenge concerns the representation of dynamic control signals (e.g. transitions or pointer drags), which must be reproduced with high fidelity for a pleasing user experience, while also minimizing network traffic and latency. Solutions might seek to downsample, approximate and compress dynamic signals ahead of distribution, and correspondingly decompress or upsample signals as part of state Assembly. Transitions can be expressed as deterministic mathematical functions, making them particularly effective in terms of network bandwidth.

Challenge 3 - Consistent state sharing: A protocol must also be defined for efficient distribution of control state and changes. Clients observing an online control source need to maintain a consistent view of control state. This may for instance be achieved by receiving the initial state on connect, followed by notifications for subsequent state changes. In a centralized architecture, global ordering for state updates can be used to ensure eventual consistency for observing clients. Different consistency models are possible for control state. A decentralized architecture might also be possible, as long as the consistency model matches application requirements.

Challenge 4 - Low latency: Control sharing over a network implies additional delays for distributing state changes. For instance, in a centralized architecture, update requests must be transmitted from client to a server, before change notifications can be multicast back to observing clients. Internet delays are generally too high for interactive applications, where immediate feedback is expected for interactive control operations. However, this can be addressed by implementing control changes speculatively locally, ahead of Internet distribution, thus removing delay for the local interface. Importantly though, speculative changes may lead to consistency issues, and occasionally be rolled back if the local ordering of state changes is not consistent with global ordering. In addition, low latency distribution is important for real-time control sharing, and collaborative scenarios in particular. General principles apply for efficient implementation of low latency distribution.

Challenge 5 - Timeline consistent control: Finally, online sharing of control state must also support timeline consistency. This means that the control state must be serialized in reference to a media clock (capture), and correctly reproduced in reference to a different media clock (playback). For example, a live control signal originating from a pointing device may be encoded as a sequence of states, each referring to a segment on the timeline. In playback, these state changes must be repeated at the correct time, including both discrete and dynamic control changes. Reproduction of control state must also be sensitive to changes to the media clock, such as pause, resume or time-shift. Timeline consistency across devices or products requires that devices and products have access to a common media clock.

B. Assembly

In control-driven media, assembly implements a conversion from data sources and control sources, into state prepared for rendering. The objective is to encapsulate application specific logic and provide simpler resource abstractions matching the requirements of render components. This way, rendering components can be efficient and stateless data sinks, as illustrated in Figure 11.

Challenge 6 - Masking heterogeneity: In CdM, assembly (blue) is represented as an independent step, decoupled from data and control sources (black, red) and rendering components (green). By encapsulating application specific functionality in the assembly step, render components may remain simple, generic, and possibly stateless. This provides opportunities for reuse of render components across different versions of assembly. Similarly, data and control sources may be reused across different assembly components, or dynamically replaced by alternative sources. To unlock this flexibility, masking heterogeneity will be a key challenge. For instance, by introducing a uniform resource abstraction, assembly functionality may work across input sources with different API or data formats.

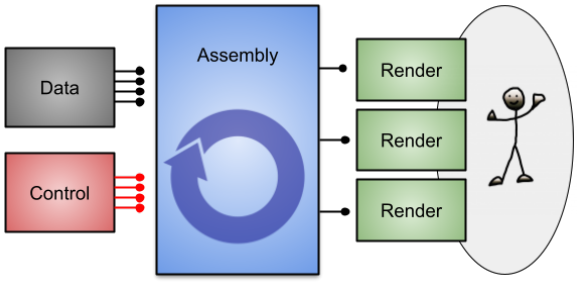

Fig. 11. Assembly converting 4 data sources and 4 control sources into 3 virtual sources of rendering state.

Similarly, if assembly exports render state in a uniform way, render components may more easily be reused across different assembly components.

Challenge 7 - Timeline consistent render state: The assembly step may also be required to produce render state in accordance with a media clock. While timeline-consistency has already been defined for control sources, it may additionally apply to sources of timed data, such as media tracks, logs or timed event sequences. For instance, given a subtitle track, the correct subtitle must be activated and deactivated in render state, during playback. Moreover, if the media clock is altered (e.g. pause, rewind, resume), or if the data source itself is modified (e.g. subtitle edit), render state must be modified accordingly. While this is feasible today using specific tools or ad-hoc solutions, in CdM though, timeline-consistency is regarded as an integral aspect of state management. The challenge then is to define a practical programming model for time-dependent state assembly, including appropriate concepts and tools.

Challenge 8 - Complexity: The assembly step implements an application specific conversion, from multiple control sources and data sources into active render state. This may involve a variety of standard processing operations such as merging, layering, filtering, aggregation, selection, transformation, etc. There may also be dependencies between sources, constraints, or even conflicts. For instance, a control source specifying more advanced graphics may have to be ignored if the screen is too small, or if power is running low. Moreover, assembly must be well-behaved for a vast set of permutations of control and data sources. This increases code complexity and the burden on developers. The challenge then is to allow application specific and dependable assembly functions to be implemented, while limiting the complexity. For instance, framework support for assembly could support advanced processing graphs, built from simple, pre-defined processing operations, each open to application-specific parametrization and customization.

VIII. Evaluation

Control and assembly are key challenges in Control-driven media (CdM). With respect to control, State Trajectory [21] has recently been proposed as a candidate solution. State Trajectory is a unifying concept for control and interactivity in data-driven applications, with built-in support for online sharing and timeline-consistent playback. State Trajectory is supported by implementation, and addresses challenges (1-5) in Section VII-A. Concepts and tools for assembly challenges (6-8) in Section VII-B is work in progress. This section presents prior research supporting CdM, and then demonstrates that CdM addresses the problem statement set out in Section III.

A. Prior Research

Control-driven media (CdM) builds on extensive research over the last 10+ years, in topics such as distributed media synchronization, real-time data sharing, multi-device media experiences, and timeline-consistent data-driven visualization. Research results include reference implementations for key concepts and services, technical evaluation, and prototype applications.

Addressing the need for timeline-consistent rendering in data-driven media experiences, the Timing Object [22] was introduced in 2015 as a generic concept for media clocks and timeline control. Timing objects are stateful resources which can be shared, observed and manipulated in application code. Research in media synchronization explored the limitation of audio and video synchronization on the Web platform, demonstrating that echoless synchronization was possible on high end smartphones as early as 2015 [23]. Sequencing tools for dynamic datasets [24] demonstrated precise synchronization of time-dependent data. Media State Vector [25] demonstrated a scalable solution for online media clocks, with global availability and precision down to a few milliseconds. Effectively, this extended the scope of Timing Objects from single interface to online multi-device applications.

Together, these concepts define a media model for timeline-consistent, data-driven, multi-device, media experiences [26]. Timingsrc [27] provides reference implementations for Timing object, Media Synchronization and Sequencing tools for the Web platform. These concepts, along with practical solutions for real-time data sharing, have consistently demonstrated their value as building blocks in data-driven, multi-device media applications.

Control-driven media (CdM) extends this model by providing a more generalized concept for control. In particular, control in CdM is not limited to media clocks, but can in principle apply to any parameter of a media experience. State trajectories [21] may represent application variables of different types (e.g. int, float, string, object, collection). Moreover, State trajectories track value changes in reference to a timeline, supporting discrete as well as time-dependent (e.g. deterministic, dynamic) changes. State trajectory has been evaluated through implementation, demonstrating a lightweight footprint (cpu, bandwidth, storage), real-time sharing, timeline-consistent playback and high potential for scaling.

Additionally, CdM extends the scope of assembly. Prior research in data sequencing [24], exclusively addressed timeline-consistency (Challenge 7, Section VII-B). In CdM, timeline-consistency is seen as part of a larger challenge - application state management. It follows that challenges regarding

timeline-consistency must be addressed alongside other software concerns, such as modularity, flexibility, composition, customization, uniformity (masking heterogeneity). To assist developers in reaching these goals, ongoing research targets framework support for assembly in CdM, and development of new and appropriate programming concepts which encapsulate support for timeline consistency.

B. Addressing the Problem Statement

1. Support consistent user experiences across platforms: CdM provides consistent user experiences across platforms by (i) representing control as an independent system component, open to integration in user facing products as well as production systems, (ii) opening up for consistent sharing of control between interfaces, and (iii) supporting timeline-consistent assembly of render state.

2. Support the combination of continuous and data-driven media: CdM allows control state to be shared between media players and data-driven interfaces (e.g. overlays, secondary devices), as a basis for consistent presentation and coordination, without introducing any additional limitations. Moreover, in CdM, control sources may be integrated with production systems for continuous media, or time-shifted for direct usage in interactive interfaces on consumer devices.

3. Support integration with existing media systems: CdM does not represent a radical break from existing systems, but rather adds control as a new and independent system component. As such, CdM remains compatible with existing data backends and rendering technologies. Moreover, the added cost of online control is likely modest, as online controls can be massively scaled by dedicated, cloud-hosted services, and do not introduce significant overhead in client interfaces [21]. A stepwise integration path is possible, starting with select render components in specific products.

4. Support high flexibility while limiting complexity: CdM maximizes flexibility by decoupling assembly from control and data sources as well as rendering components. At the same time, CdM provides powerful, generic concepts for control and assembly, encapsulating significant complexity. This combination of decoupling and unified resource representation also makes CdM a practical model for developers, with opportunities for modularity, code reuse, composition, specialization and dynamic recombination.

5. Support key trends, such as automation, personalization, collaboration: In CdM, media experiences are explicitly opened up for external control, through online control sources. This allows automated production processes to monitor live developments, and also direct a narrative through complex control sequences. CdM media experiences are highly flexible, and can support personalization of data, control sources and assembly logic. Moreover, CdM supports collaboration through real-time interactions with control and data, and also collaboration across time-frames, by time-shifting control or data.

IX. Discussion

Quality, Costs and Scalability: Quality, costs, and scalability are important measures for media providers. Importantly, Control-driven Media (CdM) does not significantly alter these measures. Quality, costs and scalability will still be largely determined by existing infrastructure for the production, distribution and rendering of data sources and media content. In comparison, control sources are shown to be light-weight entities (cpu, bandwidth, storage), and may likely be hosted cheaply at massive scale by dedicated services. On the other hand, CdM may offer improved quality of service, for instance through provider-driven interface control and consistency in presentation. Such features may be offered to premium subscribers, and motivate users to log in to sites that would otherwise be open. CdM may also support high levels of personalization at scale, while also limiting costs. This could be possible by implementing unique adaptations as part of media assembly on the client-side, thus ensuring that consumer devices carry much of the costs. CdM uniquely supports this by offering a common mechanism for production control across the entire production chain, including client-side interfaces.

Automation and AI: AI-assisted automation is a hot topic in media production, with opportunities for cheaper production of high quality content sources. However, automation of the producer role may represent an even larger potential. For instance, AI-based agents could engage in storytelling, weaving together narratives from content sources, datasets, graphical layers and interactive visualizations, adapted to individual preferences, device capabilities or other localized context. Essentially, this would correspond to a shift from text-based narratives of first generation *large language models (LLM)*, to true multimodal narration. CdM is well positioned to support this shift. Control sources provide a natural integration point, allowing AI-based agents to monitor live control state from consumers, and express narratives through manipulation of control state sequences. Moreover, AI-agents do not have to be hosted on consumer devices, but may remote control AI-driven experiences from a cloud-hosted production system. Furthermore, AI-based agents may learn directly from live interactivity, or be trained by time-shifted replays of historical sessions.

Online Multiplayer Games: CdM bears some similarity to online multiplayer games, where gameplay is driven in real-time from joysticks or fast paced point-driven interaction. Virtual 3D games also demonstrate a striking potential for client-side rendering, where hardware acceleration (GPU) is exploited to provide rich, smooth and responsive graphics, from a shared data model and a modest number of control parameters (e.g. player position, movement, orientation). However, in gaming platforms such as *Unity and Unreal* [28], [29], control mechanisms are an integral aspect of platform design, and highly optimized with respect to game logic and competitive fairness. In contrast, CdM offers technical solutions for control which may be reused in different applications, or serve as a bridge between platforms. While CdM indeed supports real-time sharing of dynamic control state, support for consistent time-shifting and playback of control may be even more valuable. For instance, in the context of live esports, CdM controls could be used to broadcast control state from competitive gameplay, opening up for slightly time-shifted rendering directly in game engines. Compared to live

video distribution, this might offer more flexibility and higher resolution, while also reducing delays and distribution costs.

Media Orchestration: CdM may also be compared to solutions for command-driven media orchestration. For instance, MIDI [30] is a classic solution for orchestration of equipment related to musical or theatrical performances. MIDI allows command messages to be broadcast within a group of devices, originally within a low-latency, cabled setup. By distributing commands regarding tempo, pitch, notes, volume etc., different instruments may be directed to play in synchrony. MIDI may be used for live collaborative sessions as well as time-shifted performances based on scripted or pre-recorded command sequences. In contrast, CdM models control as a stateful, time-dependent and generic resource, as opposed to transient, application-specific commands. This way, CdM may extend the scope of media orchestration from local to global network, from specific application domain to general usage, and also encapsulate complexity related to time-shifting.

A Unifying Media Model: Finally, CdM may be regarded as a unified model for online media, encapsulating characteristics of classical approaches, such as continuous media, data-driven, interactive media, and command-driven media orchestration. Following this, CdM helps bridging common technical divides, such as one-way broadcast vs multi-way collaboration, single vs multi-device presentation, lean-back storytelling vs lean-forward engagement, or generic coverage vs individual adaptations. With CdM these divisions can largely be re-interpreted as variations in control patterns and sharing scopes. As such, CdM may offer a unifying approach to online media. Moreover, with a more flexible foundation, media applications can avoid limitations from early design choices and easily be extended in new directions, if needed.

X. Conclusion

Vast opportunities in media technologies and automation, paired with growing viewer diversity, fuel a demand for increasingly advanced and varied media experiences. To meet the demands of both viewers and media providers, there is an increasing need to leverage complementary technologies, platforms and interfaces as part of a single, consistent user experience. While this challenge is typically addressed within the context of specific applications or platforms, this work shows that a generic solution is possible.

This paper presented *Control-driven Media (CdM)*, a new media model with built-in support for consistent, cross-platform user experiences. CdM introduces online control as a generic and independent resource type in media systems, thereby opening up for media experiences to be consistently controlled and coordinated across different device types, user interfaces, media products and technology platforms. CdM may also be seen as a generalization over existing approaches; Like continuous media, CdM supports media experiences which are *provider-driven* and *time-driven*. Like Web and native applications, CdM supports experiences which are *data-driven* and *user-driven*. Moreover, CdM is compatible with existing service backends and frontend technologies.

To further demonstrate the flexibility of CdM, the paper also presented a number of opportunities for innovation, including key challenges in the media domain, such as AI-based automation, personalization, collaboration and multi-device support. The evaluation referenced an extensive backlog of supporting research, emphasizing that solutions already exist for key technical challenges. In particular, State Trajectory [21] has recently been put forward as a candidate concept for generalized control in CdM.

Finally, by tackling the problem statement from Section III, CdM is confirmed as a practical and forward looking solution for cross-platform media experiences, with the flexibility needed to address increasingly complex demands from users and media providers.

Acknowledgment

This work was supported by the Research Council of Norway and MediaFutures (Grant number 323302 and 309339)

References

[1] W3C. (2024) HTML Living Standard Media. [Online]. Available: https://html.spec.whatwg.org/multipage/media.html

[2] (2009) Hbbtv. hybrid broadcast broadband tv. HbbTV. [Online]. Available: https://www.hbbtv.org/

[3] (2015) Dvb-css. companion screens and streams. etsi ts 103 286-1 technical specification. DVB. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/103200_103299/10328601/01.01.01_60/ts_10328601v010101p.pdf

[4] (1999) Akamai content delivery network. Akamai Technologies, Inc. [Online]. Available: https://www.akamai.com/

[5] (2009) Cloudflare content delivery network (cdn). [Online]. Available: https://www.cloudflare.com/cdn/

[6] (2011) Fastly content delivery network. [Online]. Available: https://www.fastly.com/

[7] ISO, "Dynamic adaptive streaming over HTTP (DASH)," International Organization for Standardization, Standard, Aug 2022. [Online]. Available: https://www.iso.org/standard/83314.html

[8] R. Pantos and W. May, "Http Live Streaming," Internet Requests for Comments, IETF, RFC 8216, August 2017. [Online]. Available: https://tools.ietf.org/html/rfc8216

[9] I. Fette and M. Alexey., "The WebSocket Protocol," Internet Requests for Comments, IETF, RFC 6455, Dec 2011. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6455

[10] (2015) Object-Based Media. BBC Research & Development (R&D). [Online]. Available: https://www.bbc.co.uk/rd/object-based-media

[11] M. Armstrong, M. Brooks, A. Churnside, M. Evans, F. Melchior, and M. Shotton, "Object-based broadcasting-curation, responsiveness and user experience," in *IBC2014 Conference*. IET Digital Library, 2014.

[12] M. Evans, T. Ferne, Z. Watson, F. Melchior, M. Brooks, P. Stenton, I. Forrester, and C. Baume, "Creating object-based experiences in the real world," *SMPTE Motion Imaging Journal*, vol. 126, no. 6, pp. 1–7, 2017.

[13] (2021, Sep) Object-based Media report. The Office of Communications (Ofcom). [Online]. Available: https://www.ofcom.org.uk/research-and-data/technology/general/object-based-media

[14] (2018) Formula 1 TV (F1TV). [Online]. Available: https://f1tv.formula1.com/

[15] (2018) Formula 1 Mobile App (F1 App). [Online]. Available: https://www.formula1.com/en/subscribe/download-the-official-F1--app.html

[16] "Document Object Model (DOM)," Tech. Rep., Feb 2024. [Online]. Available: https://dom.spec.whatwg.org/

[17] C. Brittle. (2023, 11) F1 2023 season review: Steady tv viewership, attendances rising, and the andretti problem. Blackbook Motorsport. Accessed: 2024-03-11. [Online]. Available: https://www.blackbookmotorsport.com/features/f1-2023-season-review-tv-viewership-attendance-andretti/

[18] (2022) Formula1. stream f1 live, your way. Accessed: 2024-03-11. [Online]. Available: https://www.formula1.com/en/subscribe-to-f1-tv.html

[19] (2024) Feature: Data and electronics in f1 explained. AMG Petronas Formula One Team. Accessed: 2024-03-11. [Online]. Available: https://www.mercedesamgf1.com/news/feature-data-and-electronics-in-f1-explained

[20] M. Zorrilla, N. Borch, F. Daoust, A. Erk, J. Flórez, and A. Lafuente, "A web-based distributed architecture for multi-device adaptation in media applications," *Personal and Ubiquitous Computing*, vol. 19, pp. 803–820, 2015.

[21] I. M. Arntzen, N. T. Borch, and A. Andersen, "State Trajectory. A Unifying Approach to Interactivity with Real-Time Sharing and Playback Support," in *Proceedings of the Future Technologies Conference (FTC) 2023, Volume 2*, K. Arai, Ed. Cham: Springer Nature Switzerland, 2023, pp. 1–20.

[22] I. M. Arntzen, F. Daoust, and N. T. Borch, "Timing Object; Draft community group report," Tech. Rep., Nov 2015. [Online]. Available: http://webtiming.github.io/timingobject/

[23] N. T. Borch and I. M. Arntzen, "Mediasync Report 2015: Evaluating timed playback of HTML5 media," Norut Northern Research Institute, Tech. Rep. 28, Dec 2015. [Online]. Available: https://hdl.handle.net/11250/2711974

[24] I. M. Arntzen and N. T. Borch, "Data-independent Sequencing with the Timing Object: A JavaScript Sequencer for Single-device and Multi-device Web Media," in *Proceedings of the 7th International Conference on Multimedia Systems*, ser. MMSys '16. New York, NY, USA: ACM, 2016, pp. 24:1–24:10.

[25] I. M. Arntzen, N. T. Borch, and C. P. Needham, "The Media State Vector: A Unifying Concept for Multi-device Media Navigation," in *Proceedings of the 5th Workshop on Mobile Video*, ser. MoVid '13. New York, NY, USA: ACM, 2013, pp. 61–66.

[26] I. M. Arntzen, N. T. Borch, and F. Daoust, "Media Synchronization on the Web," *MediaSync: Handbook on Multimedia Synchronization*, pp. 475–504, 2018.

[27] I. M. Arntzen. (2015) Timingsrc. Multi-device Timing for Web. [Online]. Available: https://webtiming.github.io/timingsrc

[28] (2005) Unity real-time development platform. Unity Technologies. [Online]. Available: https://unity.com/

[29] (1998) Unreal engine. Epic Games. [Online]. Available: https://www.unrealengine.com/

[30] (1985) Midi. musical instrument digital interface. [Online]. Available: https://midi.org/