Scheduling Interactive and Concurrently Running Applications in Pervasive Display Networks

Ivan Elhart¹, Marc Langheinrich¹, Nemanja Memarovic¹, and Tommi Heikkinen²
¹Faculty of Informatics University of Lugano (USI) Lugano, Switzerland {firstname.lastname}@usi.ch
²Department of Computer Science and Engineering University of Oulu Oulu, Finland {firstname.lastname}@ee.oulu.fi

ABSTRACT
Today’s digital signage systems typically show advertisements in the form of images or short videos that their owners and/or operators have arranged in well-defined sequences well before presentation time. In order to make such displays more attractive, both researchers and advertisers have recently begun to explore the concept of interactive applications that allow passers-by to directly or indirectly control a display’s content. To integrate such applications with traditional digital signage concepts requires new forms of shared control over application and content scheduling, ultimately creating multi-user and multi-application display systems that go beyond predefined sequences of content items. In this paper we present a system for scheduling both interactive content and traditional digital signage content on networked public displays. We offer a formal notation for describing such novel scheduling problems, based on a list of requirements for scheduling interactive and concurrently running display applications, and describe a web-based application development framework and API for dynamic application scheduling. We also report on an initial prototype system that we have deployed on a university campus.

Categories and Subject Descriptors
H.5 [Information Interfaces and Personalization]: Miscellaneous

Keywords
Public displays, interactive applications, scheduling.

1. INTRODUCTION
Public displays are becoming a ubiquitous resource in the urban environment. Most of these displays are designed around a digital signage model that provides content control and scheduling mechanisms only to display owners – individuals or organizations that physically own or operate public displays. In this model, display content is first scheduled in a well-defined sequence of content items (images or videos) and then distributed to displays for presentation [11]. Digital signage displays are usually non-interactive and are often ignored by passers-by [13].

Recent work in public display research has proposed a number of interactive systems to better “connect” with audiences [2], [10], [14], [15]. These systems allow passers-by to not only influence content display, but in some cases to also upload and show their own content. Opening public displays up to viewers with the help of interactive applications can make public displays more attractive and more appreciated in their environment [4]. However, integrating such interactive applications, let alone user-contributed content, with the pre-planned and carefully curated content playlists in a digital signage model is a challenge, as display owners and viewers often have differing requirements [1].

In our work, we aim at providing new forms of shared control over interactive public displays, integrating display owners, display viewers, and individual applications. Therefore, our main research question is: how can we open today’s closed and isolated public displays to third party applications and user-contributed content, but still keep their owners in overall control? In a very simple scenario, display owners could decide how much display resources (time and space) they can offer to interactive applications and display viewers and allow the display system to automate the scheduling decisions. While there are different social and economic issues surrounding this vision [4], here we are looking into the technical challenges of scheduling applications on public displays. We build upon our initial work previously in various workshops and poster sessions, in particular on identifying key scheduling challenges [6], adapting a scheduling notation for public displays from general scheduling theory [8], and providing a display control interface for display owners [9].

In this paper, we present a comprehensive approach to scheduling both interactive applications and traditional digital signage content on networked public displays. First, we present a set of common application scheduling requirements that we obtain from both display owners and display viewers. Second, we present a formal notation for describing such scheduling problems in order to construct scheduling policies for public displays. Third, we present a web-based application development framework and a set of scheduling API calls that concurrently running applications can use. Finally, we report on a prototype display system that we fielded in a university context.

2. RELATED WORK
A number of projects have previously suggested the use of both interactivity and user-contributed content in order to increase viewer participation, such as Instant Places [10], Screens In The Wild [14], UBI-hotspots [15], and e-Campus [2]. We complement on the work on existing interactive display systems by presenting a modular approach to scheduling traditional signage content with concurrently running interactive applications that takes into account the preferences of display viewers and display owners.

Instant Places is a display platform that features a number of web-
based display applications such as a “presence” application that shows the profiles of users around the displays and their interests in form of small “pins”, or a “poster” application for publishing user contributed posters [10]. These applications are scheduled on the displays in a traditional digital signage way: display owners first arrange applications in time sequences and then upload a fixed schedule to the displays. This approach allows only display owners to influence an application’s start and presentation times. While display viewers can personalize individual applications, they have to wait for the applications to appear on the display and they cannot influence the application presentation time.

The Screens In The Wild project features a network of four public displays integrated into the urban environment: a public library, a community center, an art space, and a cinema [14]. The screens show interactive applications such as VideoLink, SoundShape, ScreenGram, and Moment Machine [12]. A dedicated scheduling component switches between the applications on predefined time intervals and shows each application on all four screens at the same time. Similarly to Instant Places, display viewers can personalize individual applications (e.g., take pictures with a display-mounted camera), but cannot influence the application start and duration times.

UBI-hotspots is a network of touch-enabled indoor and outdoor public displays installed in the city center of Oulu, Finland [15]. The displays combine standard digital signage content (images and videos) with interactive applications through a state machine with two defined states: passive and active. In the passive state the displays show digital signage content arranged in a sequence and shown in a full-screen mode called UBI-Channel. In the active state the screen is divided into two areas, one showing the UBI-channel and one offering a number of interactive applications, called UBI-portal. This approach separates concerns of display owners, giving them full control over UBI-channel, and display viewers, giving them full control over UBI-portal. While there are certain advantages to this separation, there are also some possible limitations. First, display owners do not have full control over their displays, e.g., to show high priority content in full screen in certain periods of time. Second, display viewers cannot gain control over the entire screen area even for short interactions.

e-Campus is a university wide installation of public displays also featuring a number of interactive applications such as interactive map, Flicker photo view, YouTube video view, art installations, and applications that show university-wide and location-specific content, among others [2]. The e-Campus platform provides flexible ways for developing independent constraint-based schedulers using low-level and high-level API calls. It supports the existence of multiple independent schedulers at the same time, but it provides no mechanisms for describing the actual behavior of such schedulers on the final application presentation times. The platform offers a standalone display player that uses so-called Content Descriptor Sets for describing content items and instructions on how and when they should be displayed [3].

3. SCHEDULING CONCURRENT AND INTERACTIVE APPLICATIONS

Application scheduling on public displays can be defined as a decision making process that deals with allocating display applications to one or more displays or display regions, as well as presenting a particular content item within an application. This decision making process directly controls what content is shown and when it is shown on the screens.

Application scheduling is thus one of the central challenges for opening public displays to interactive applications, and can be seen as the point where four principal sources of requirements come together: a) preferences of display owners, b) interests of display viewers, c) applications and their content, and d) a display’s physical and social environment [6]. The interplay of such different parameters makes scheduling interactive applications on public displays a complex problem.

3.1 Scheduling Requirements

In scheduling applications and their content on public displays, a distinction can be made between traditional (static) content and interactive applications [6], [9]. We extend this set of requirements from our previous work with additional requirements of scheduling concurrently running applications.

3.1.1 Traditional content

In digital signage, content is typically scheduled well before final presentation time. Traditional content is usually associated with scheduling requirements in the form of scheduling constraints:

- "show item1 from 14:00 to 15:00",
- "show item2 n number of times",
- "do not show item3 after 20:00",
- "do not show item4 more than n number of times", or
- "show item2 immediately after item4"

The scheduling constraints are processed by a constraints solver or manually by display owners using a display scheduling software. The output of the software is a sequence of content items, usually called a “playlist”. The items in a sequence can be additionally fine-tuned to produce the desired display behavior. Such playlists are usually produced at a central location and distributed to displays for final presentation.

Having the finite set of content items arranged in a well-defined sequence has two important implications. First, the start times of all content items are known at any moment of the presentation time. Second, the duration of all content items is fixed during the entire presentation time. In contrast, interactive applications usually do not have these properties and traditional scheduling approaches cannot be applied to them.

3.1.2 Interactive content

Scheduling interactive applications is more complex than scheduling traditional content. Interactive applications do not have the properties of static content, creating two main problems:

1. **Arbitrary application start time**: Scheduling content in traditional systems is based on the assumption that the start times of all content items are known in advance. In open display networks, applications can be triggered at arbitrary times, e.g., when passers-by touch a display. Scheduling interactive content must support arbitrary start times.

2. **Arbitrary application duration**: Digital signage systems assume that content items are of fixed duration. However, interactive applications can be of arbitrary duration depending on actual interaction. Removing an application while users are interacting with it negatively influences the user experience and the overall usability of the system.

Such a dynamic environment requires new approaches to scheduling interactive applications on public displays that go beyond a predefined sequence of content items [6].

3.1.3 Concurrently Running Applications

Traditional content is usually presented in a full-screen mode using time multiplexing techniques. However, recent public installations have started exploring space multiplexing together
with time multiplexing. For example, a display could be divided into three concurrent zones – two-thirds main area, one-third sidebar, bottom tickertape – or be used full-screen [11].

In our work we assume that applications can present content not only on one or more displays but also on one or more “screen zones” (different sizes and resolutions). Presentation of application content on multiple displays is mainly concerned with synchronization of multiple instances of the same application (application clients). However, when a display is divided in different screen zones, multiple concurrent applications can run and show their content at the same time, potentially interacting with each other. We can define four types of such applications [6]:

1. Applications unaware of each other (independent applications). They concurrently compete and negotiate for display resources and independently schedule content items for presentation in one or more display regions.

2. Applications indirectly aware of each other. They share access to a common resource such as local sensor input. They independently make decisions to show content, but the shared resource may influence their decisions. This may result in showing related content in different regions at the same time.

3. Applications directly aware of each other. They directly communicate and cooperate in presenting content. An application can request another application, potentially passing parameters for visualization. These applications do not have to be visible at the same time.

4. Applications dependent on each other. They directly communicate and cooperate in presenting content. A directly dependent application requires all dependent applications to be simultaneously visible in the nearby regions in order to create desired user experience.

Supporting concurrently running applications has several advantages. First, it enables individual applications to request display resources when new and relevant content becomes available. Also, it enables individual applications to request presentation time for showing highly personalized content to display viewers [5]. Second, it enables easier development of menu type applications, allowing users to directly request applications for presentation. Third, it allows development of specialized coordinated multi-zone applications such as Memarovic et al.’s “Moment Machine” [12].

3.2 Scheduling Notation for Public Displays

Digital signage usually defines a scheduling problem with a set of scheduling constraints associated with content items. In order to integrate interactive applications with traditional content and to describe dynamic behavior of future public displays, a more general scheduling notation is needed.

3.2.1 General Scheduling Theory

In general scheduling theory, scheduling can be defined as “a decision making process that deals with allocation of resources to given tasks over a certain period of time with the main goal of finding a feasible or optimal solution that is subject to a given set of constraints and one or more objectives” [16]. A mathematical representation of a scheduling problem is usually given as a sequence of three elements: \( \alpha \mid \beta \mid \gamma \), where \( \alpha \) describes a machine environment and available resources, \( \beta \) provides information about tasks and associated constraints, and \( \gamma \) describes an objective to be optimized [16].

The machine environment describes the number of available machines and their mutual relations and possible processing constraints. Some of the common machine environments include parallel machines with different processing speed, flow shops, job shops, or open shops. The tasks and constraints field holds information about individual tasks and constraints associated with them. One can distinguish between hard and soft constraints as well as static and dynamic constraints. In a scheduling problem, hard constraints have to be satisfied at all cost. In contrast, soft constraints are usually described as preferences of lower priority. While static constraints are constant during the processing time, dynamic constraints can change over time and depend on other constraints. The overall goal of scheduling is to optimize one or more objective functions given the machine environment, tasks, and associated constraints. The common optimization techniques are concerned with finding an optimal or feasible solution that minimizes the objective function, such as total completion time.

In a public display domain, objectives typically include profit from advertisers but also user satisfaction. These objective functions might be very simple or very complex to construct.

3.2.2 Notation for Public Displays

Following the notation above, we propose to describe the scheduling of applications on public displays as a set of three elements, \( \alpha \mid \beta \mid \gamma \), that describe: \( \alpha \) - a display environment and screen real-estate division, \( \beta \) - an application environment and associated application presentation constraints, and \( \gamma \) - a set of heuristic rules that define the overall display behavior (instead of optimization functions).

The display environment \( \alpha \) describes the number of displays, their mutual relations, and their respective division of screen real estate. For a particular scheduling problem and a given set of applications, the relations among a set of public displays can be:

- Independent displays (D): Displays show independently one or more applications that are available for scheduling.
- Displays in series (S): An application can use displays in series to enable its content to “travel” from one display to another in a time-defined sequence, e.g., for information that follows users as they walk through a corridor or hallway.
- Displays in parallel (P): An application can use displays in parallel to show the same content on all displays during a specific time interval, e.g., emergency information.

In addition to describing the display environment, the display environment \( \alpha \) can also contain information about the screen real estate division. Screen zones supported in our system include:

- Full-screen (f): an application takes the entire screen area.
- Sidebar (s): an application has access to a small area to show, e.g., a preview of its content until it gets full-screen access.
- Main area (m): an application can show content on the screen area that is left (e.g., two-thirds) when the sidebar is visible.
- Tickertape (t): an application can show content in a small area at the bottom of the screen.

The screen real estate division is not limited to these four elements. It is not hard to imagine that display owners may want to define different and unique screen zones for their displays.

The second element, the application environment \( \beta \) describes the applications and their associated constraints. This includes a list of applications that can be scheduled on the display, constraints about the applications’ presentation and duration times, as well information about the association of applications with particular display zones. Some of the most common scheduling constraints that can be applied to public displays are:

- Processing time (p): represents the limits of the application presentation time. It can be used to set both a minimum and
maximum presentation times.

- Release time (r): represents the start time of an application. It can be expressed in absolute times, times relative to other applications, or as time intervals (recurring).
- Due time (d): represents the envisioned/desired completion time of an application. It is possible that an application completes its presentation before or after its due time.
- Weight (w): a “priority factor” that gives a higher priority of presentation to applications with higher weights. Note that this can lead to “application starvation”, where low-priority applications cannot access the screen due to the constant presence of high-priority applications.
- Pre-emptions (prmp): allows the scheduler to interrupt and stop the presentation of an application at any point in time to allow an application with higher priority to access the screen. This is a boolean flag that can be applied to individual applications or all running applications as a display rule.
- Precedence (prec): denotes the requirement that one or more applications have to be completed before a particular application can start its presentation.

The number of applications, as well as their corresponding scheduling constraints, can change over time, allowing for describing a more dynamic behavior of displays. For example, the weights associated with displays applications can change according to the time of day or the presence of passers-by. Note that pre-emptions are of central importance for scheduling interactive applications with arbitrary start times. It is important that display applications can start their presentation with minimal delay when users request them. However, when an application is substituted with another application, the system should provide visible clues for the cause of change.

The third element of the scheduling problem, the heuristic rules $\gamma$, defines the overall display behavior. Rules can be static or dynamic. Composite rules are combinations of elementary rules in the form of a ranking expression. Elementary rules may include:

- **Priority based (PB)** - where each application is assigned a priority and the rule always chooses an application of the highest priority. However, if there is more than one application of the same priority, the algorithm may use an additional priority policy to make the decision.
- **First-come-first-served (FCFS)** - where applications form a queue of ready applications waiting for the airtime. When the current application finishes its presentation, the application that has been in the queue the longest is selected for presentation.
- **Round Robin (RR)** - or time slicing, where the rule makes the decision which application to show at periodic intervals. At the end of the interval, the algorithm pre-empts currently running application and selects the next application according to additional scheduling criteria. The duration of the time interval is the main issue for this type of algorithms. Since interactive applications are of different duration, a short time interval may often interrupt the user interaction.
- **Default Activity (DA)** - where the display switches to a default rule or default application (set of applications) after a certain time interval of no user activity on the display. This rule can be used to model transitions between active and passive modes such as in the UBI-hotspots [15].

We use the scheduling notation to construct scheduling policies, based on the requirements of display owners, which directly influence the behavior of a display scheduler (section 4.3). A scheduling policy is similar to Clinch et al.’s content descriptor set in the e-Campus system [3]. However, there are two differences. First, the scheduling policy consists of three parts that strictly follow the scheduling notation described above. Second, while the content descriptor set is comprised of content-sets and content-items, the scheduling policy is concerned only with scheduling applications that contain one or more content items.

### 3.3 Scheduling API

In order to support concurrently running applications and enable them to independently request display resources, our scheduling framework supports three core API calls that an application can make to the central scheduling process running on a display:

- requestApplication(appName, when) : response
- requestApplicationInfo(appName) : response
- subscribeForApplicationInfo(appName) : response

#### 3.3.1 Requesting Applications

An application can request another application for several reasons:

a) to continue item presentation, e.g., a picture taking application passing a picture to a picture gallery application, b) to give air time to other apps in case the application does not have new content of its own, and c) to start an application selected by viewers (invoked by a menu type application).

```plaintext
requestApplication(appName, when) : response
```

The parameter appName indicates the requested application, while when specifies the time interval when the application is desired for presentation. The when parameter supports two types of requests:

- “now”: requesting an immediate application display.
- “from hh:mm to hh:mm”: an application can request an application in a future time interval, e.g., to take the first available time slot for presentation and inform viewers when the application will be shown.

An application that sends such a request to a scheduler can get one of three possible responses:

- “yes” – the application is available and will be shown as desired (“now” or in requested time interval).
- “no” – the application is not available in the requested interval during the processing time (e.g., a calendar day) due to other scheduling requirements of higher priority.
- “no, but can be available from hh:mm to hh:mm” – the requested application is not available as desired, but there is an available slot outside the requested period. If desired, applications can thus send an updated request.

#### 3.3.2 Obtaining Scheduling Information

Prior to scheduling other applications on a public display, an application can request scheduling information about other applications. Applications can request scheduling information from the scheduler in three possible ways:

- Information about all available applications, e.g., to know other running or potentially running applications.
- Information about a particular application, allowing an application to obtain scheduling information about a desired application before requesting it for scheduling.
- Updates about all applications available for scheduling, allowing an application to get changes in scheduling decisions.

To request information about all applications, an application uses the parameter appName = “all” in the API call below:

```plaintext
requestApplicationInfo(appName) : response
```
In this way, a menu application could obtain information about applications and update the menu accordingly. The response is a list of: application names, a URI to the application icon, and a set of time intervals when the application is available for scheduling. If a specific application name is used, the response instead is simply the set of time intervals when the requested application is available for scheduling. A part of the message also indicates if the application is currently visible on the screen.

The time intervals when an application or a set of applications are available for scheduling can change during the run time of the scheduler, due to the dynamic way of making decision when to schedule applications. Therefore an application can subscribe to application scheduling info to get updates when an application’s availability for scheduling changes:

```
subscribeForApplicationInfo(appName) : response.
```

4. Implementation - Prototype System

We have implemented our scheduler as part of a larger open public display application framework called “WE-BAT” [7], [8]. Below we will describe the general framework (section 4.1) as well as its scheduling component (section 4.3).

4.1 Application Development Framework

WE-BAT is a web-based client-server architecture shown in Figure 1. The display framework is based on the Java PLAY client-server framework (http://www.playframework.com/). WE-BAT consists of three main components: 1) hooks to online platforms or local folders that allow easy collection and publishing of information, 2) an application server that provides presentation. All communication between the clients and the server is achieved through JSON messages.

Each application client has a unique ID allowing content distribution to a single client, a group of clients, or all available clients (broadcast mode). Application clients receive messages or new content items form an application server. Clients typically provide additional application logic that supports interactivity through inputs from local sensors (e.g., touch events, or local camera feed), and optionally support different content layouts on a display. Clients are written using HTML, JavaScript, and CSS. They run in a virtual display player (a standard web browser page) that contains an instance of the display scheduler (section 4.3).

4.2 Types of Display Applications

According to the design requirements for developing and scheduling applications we distinguish five application types:

1. **Sourcing applications** bring user generated content from online social media to public displays. An example is the Twitter application that sources tweets with a specific hashtag or those that mention a specific user or topic.

2. **Situated applications** allow passers-by to contribute content directly to a display, e.g., by taking a picture of themselves with a camera attached to the screen for an on-screen album.

3. **Posting applications** allow viewers to send content from a display to a social network, e.g., the previously mentioned pictures could be sent onward to a public Facebook account.

4. **Signage applications** show traditional signage content (images and/or videos) – either non-interactively in full-screen mode or in an interactive fashion, organized around subtopics that display viewers can browse on the screen.

5. **Introspection applications** contain information about the available application on the screen and when and how they can be scheduled/shown on the screen, depending on the scheduling constraints. They allow for interactive menu-style applications, or for alerting users to upcoming time slots that block interactivity for full-screen display signage content.

Different types of applications come with different scheduling requirements. Sourcing applications should be able to request display resources and present new content when it becomes available. Presenting content when it is fresh can increase the relevance of the displays in their environment. Posting applications and situated applications should be available when display viewers request them at the moment of interaction. Display owners might want to show Signage applications in a non-preemptive way during predefined time intervals.

4.3 Display Scheduler

The display scheduler runs in a standard web browser that supports websocket connections. The scheduler divides a web page into supported screen zones, provides support for application scheduling constraints, and controls application presentations according to display rules. The input to the scheduler is a scheduling policy by display owners and scheduling requests by both display applications that implement the scheduling API (section 3.3) and display viewers through introspection applications (section 4.2). The scheduling policy is stored in a database as a set of tables that describe the scheduling problem, following the scheduling notation described above, and distributed to individual screens in JSON format. The scheduler has a modular architecture that can be extended with additional constraints and display rules. The structure of the scheduler is shown in Figure 2.
4.4 Deployment
In order to test and experiment with different ways of scheduling concurrently running applications, we have deployed our scheduler as part of a public display system in our university [8] that has been in daily use since early February 2014.

Using the scheduling notation described above, we can describe the scheduling problem that underlies the developed system as:

\[ D_{i=1,...,4} L_1 = \{ f, m, s, tt \}; D_{i=1,...,4} \rightarrow L_1 \]  
\[ a_{j=1,...,17}; \ a_{1,...,17} w(t); \ a_{6,...,17} d(t); \ a_{1,...,17} \rightarrow \{ s \}; \ a_3 \rightarrow \{ t; \ a_4 \rightarrow \{ m \}; \ a_{6,...,17} \rightarrow \{ f, m \}; \]  

\[ RR(t), DA(t), PRMP(t) \]

The deployed systems consists of four displays \( D_{i=1,...,4} \) deployed in two university buildings. All displays have the same layout that consists of four screen zones, i.e., full-screen, sidebar, main area, and ticker tape. Each display shows a set of 17 applications that can present their content in one or more screen zones indicated by the application constraints: a Twitter app (\( a_1 \)) – a sourcing display application that enables display viewers to send tweets to the displays; the Moment Machine app (\( a_4 \)) – a situated and posting application that allows display viewers to take in situ images through a webcam and publish them on the screen or on a dedicated Facebook page; the Moments Gallery app (\( a_5 \)) – a situated application that shows images taken by the Moment Machine app; a set of 15 signage applications (\( a_{6,...,17} \)) that show institutional content (e.g., Academic Calendar, Career Opportunities, Housing Info) in the form of slides organized into subcategories; and two introspection applications (\( a_{3,5} \)) that allow viewers to request available applications via a small selection bar and a larger “cover page” view.

The display behavior is defined by three rules. The screens show applications using round robin (RR) when there are no viewers interacting with the display through a touch screen interface and when the display is in pre-emptive state. Display owners can control pre-emptive behavior of the display (PRMP), in other words, lock the display to a subset of applications. After a time interval (timeout) of no interactivity the screen will resume default activity (DA), i.e., showing applications in RR fashion.

5. CONCLUSION
Traditional digital signage content and interactive applications pose different scheduling requirements for public display systems. In this paper we presented a scheduling notation used for describing scheduling problems and constructing scheduling policies for public displays, as well as a sample implementation of such a scheduler in form of a modular web-based public display system that integrates interactive applications with digital signage.

We are currently in the process of collecting both performance and interaction data in a campus-wide real-world deployment, in order to better understand limitations and capabilities of the system, its actual use, and corresponding stakeholder needs.

6. REFERENCES