Are You Feeling Lucky? : Lottery-based Scheduling for Public Displays

Mateusz Mikusz, Sarah Clinch and Nigel Davies
School of Computing & Communications
Lancaster University, Lancaster, UK
m.mikusz | s.clinch | n.davies @ lancaster.ac.uk

ABSTRACT

Scheduling content onto pervasive displays is a complex problem. Researchers have identified an array of potential requirements that can influence scheduling decisions, but the relative importance of these different requirements varies across deployments, with context, and over time. In this paper we describe the design and implementation of a lottery-based scheduling approach that allows for the combination of multiple scheduling policies and is easily extensible to accommodate new scheduling requirements.

Author Keywords

public displays; scheduling.

ACM Classification Keywords

H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

INTRODUCTION

Content scheduling remains a key challenge for emerging pervasive display systems. Traditional digital signs feature relatively static display schedules expressed as simple deterministic playlists [6]. However, as display systems evolve to embrace a wide range of content types and stakeholders the scheduling complexity increases significantly [18, 10].

To address the problem of flexible content scheduling researchers have proposed the use of constraint-based schedulers [12] in which content items are tagged with a series of constraints (e.g. only show a specific content item between 9am and 5pm) that help determine when each item can be presented. Unfortunately, *the use of constraints only provides a partial solution to the scheduling problem*. It is *necessary* to understand the constraints governing when content items can be shown but this is *not sufficient* to determine which item to show at a given time as there are likely to be multiple content items that satisfy the current constraints. Similarly, constraints are not well suited to expressing the rich mix of priorities and policies that may further influence content selection.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
PerDis '15, June 10 - 12, 2015, Saarbruecken, Germany
Copyright © 2015 ACM. ISBN 978-1-4503-3608-6/15/06...$15.00
DOI: http://dx.doi.org/10.1145/2757710.2757721

For example, display owners may wish to show a particular mix of content types, optimise for different content lengths or increase the probability that content relevant to viewers is shown when they are close to a display. Simply adding priorities to constraints is not sufficient as priority levels typically fail to provide the fine-grained levels of control required.

Consequently, future pervasive display systems will require scheduling techniques that extend beyond simple constraint or priority based systems and there is a pressing need for new insights into the problem of content scheduling [9]. In this paper we explore the use of lottery scheduling [19] as a technique for scheduling content in pervasive display networks. Originally conceived for use in operating systems, lottery scheduling provides an efficient means of achieving proportional-share scheduling, with probability ensuring fairness and preventing starvation. Scheduling priorities and policies can be implemented through changes in ticket allocations that are immediately reflected in subsequent draws, hence allowing lottery scheduling to be responsive to change (e.g. for interactive systems).

Building on over twenty years experience of creating display scheduling systems we describe how lottery scheduling can be applied to the pervasive display domain and then present the design, implementation and evaluation of the system in a live signage network. While lottery scheduling does not represent the total solution to scheduling on pervasive displays, our experiences suggest that it provides an effective way of addressing many of the most challenging forms of scheduling requirements and can be combined with other scheduling techniques as part of a comprehensive solution.

LOTTERY SCHEDULING

Lottery scheduling was first described in the operating systems literature by Waldspurger and Weihl [19] and is a probabilistic mechanism for allocating resources.

A lottery scheduling algorithm allocates ranges of tickets (representing resource rights) to competing clients (e.g. to determine which process will execute). Following ticket allocation, a draw is held to determine the winning ticket and thus which client receives the resource. In this way, lottery scheduling provides a random but fair means of achieving proportional-share scheduling (i.e. each item is scheduled according to its allocated proportion of tickets). Furthermore, since changes in ticket allocations are immediately reflected in the next draw, lottery scheduling quickly responds to changes in resource rights.

In their original description of lottery scheduling, Waldspurger and Weihl provide three additional mechanisms for altering the proportions of tickets allocated to each client. *Ticket transfers* allow a client that cannot make use of a resource to reallocate its tickets to another—for example, a process waiting on a lock may transfer its tickets to the process currently holding the lock in order that the lock may be released more quickly. *Ticket inflation* allows a client to escalate its own resource rights by creating more lottery tickets for self-allocation. *Compensation tickets* allow a winning client that yields use of the resource to be granted additional tickets in future draws in order to maintain correct proportional use of the resource.

Finally, lottery scheduling can also be used with a currency abstraction that enables load insulation between groups of clients. Specifying the exchange rate between "currencies" enables complex allocations to be carried out with relative simplicity.

APPLICATION TO PUBLIC DISPLAYS

Lottery scheduling was originally proposed for use in operating systems. However, we believe that the same technique can be effectively employed in pervasive display networks and in the following sections we illustrate how lottery scheduling can be used to meet typical display scheduling requirements.

Ratio Based Scheduling. A common scheduling requirement in display systems is to be able to schedule content at different ratios. For example, a display owner might want to control the balance of news, adverts and music videos on a display. In a system based on lottery scheduling this simply means allocating the tickets in the appropriate ratio. Clearly some care is needed when dealing with content items of different lengths and the following formulae produces the expected results:

$$\text{proportion of tickets to allocate}_i = \frac{\text{timeshare}_i \cdot \sum_{j=1}^{N} \text{duration}_j}{\text{duration}_i} \cdot \left(\sum_{k=1}^{N} \left(\frac{\text{timeshare}_k \cdot \sum_{j=1}^{N} \text{duration}_j}{\text{duration}_k} \right) \right)^{-1}$$

Prioritisation of New Content. When signage systems have a large quantity of possible content to select from it is often important to be able to increase the likelihood of fresh content being shown. This requires ticket allocations to be adjusted such that new items of content receive a disproportionate allocation of tickets – with this allocation reducing over time.

Reflecting Viewer Linger Times. The amount of time viewers spend in front of a display may be subject to significant variation. For example, in a work cafe viewers may just glance at a display in the morning when they collect a coffee but then have longer to look at the display when eating their lunch. Ticket allocations could be varied to help increase the probability of content of an appropriate duration being shown at a given time. While this kind of tailoring could be implemented as constraints (i.e. only show content of an appropriate length during a specific time slot) the use of ticket allocations provides a probabilistic approach that enables displays to subtly adjust their schedules over time to match viewer behaviours.

Long term personalisation. Display personalisation systems such as Tacita [8] offer the potential for users to influence the content shown on displays through both walk-by and longitudinal personalisation. The latter form of personalisation, in which users influence the long term trend in content scheduling, can be well supported using lottery scheduling through allocating tickets based on the presence of viewers.

These examples illustrate how lottery scheduling can be used to support a range of common scheduling requirements. Crucially, these are representative of requirements that are particularly difficult to support using standard constraint or priority schemes as they require fine grained control over the long term probability of content items being scheduled rather than an immediate response to changes in constraints or priorities.

However, it is important to stress at this point that we are not proposing lottery scheduling as a complete solution to display scheduling. Rather lottery scheduling appears applicable as part of a scheduling eco-system that includes an appropriate mechanism (typically a UI) to enable the user to specify their scheduling requirements and support for constraints that can be processed prior to consideration in the context of a lottery.

For example, once the user has specified the content items and scheduling constraints the overall sequence of operations could be: 1) process constraints to produce a set of content items that could be presented, 2) allocate tickets to these content items (or groups of items), and 3) perform a draw to determine the "winning" content item. This sequence of operations is repeated every time a new scheduling decision needs to be made.

An additional benefit of lottery scheduling is that the above examples of ticket allocation can be combined in multiple ways. Thus it is possible, for example, to combine a ticket allocation strategy that priorities new content with one that supports long-term personalisation without significant additional complexity. It is also possible to run multiple lottery draws prior to scheduling a content item. This can be useful, for example, to enable the use of an initial round of lottery scheduling to determine which set of content items to show and then to run an additional draw within the set to determine the exact content item to present. Different ticket allocation strategies can be employed for each draw.

DESIGN AND IMPLEMENTATION

Design Overview

We propose a generalised architecture for a digital signage lottery scheduler based on six components: a *scheduling manager*, a *context and constraints parser*, a *context store*, a *filter pipeline*, a *lottery scheduler*, and a *configuration* component (Figure 1).

The scheduler operates on an initial input of content items and their associated constraints. The scheduling process is orchestrated by the manager and involves the following sequence of steps. First, the context and constraints parser processes the content items and constraints into a standard format understood by the remaining components. The filter pipeline then removes content items that cannot be played on

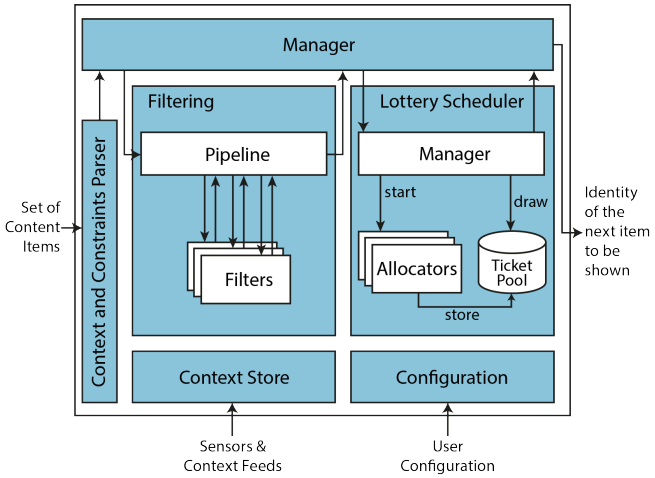

Figure 1: Lottery scheduler architecture.

the screen from the available set. The filtered content items are then passed to the lottery scheduler and ticket allocators for the distribution of tickets and subsequent draw. Finally, the identity of the winning content item is produced as the output of the system. All components have access to a shared context and configuration store. We now describe each of these components in more detail.

Scheduling Manager

The manager component coordinates the overall scheduling process—waiting for new updates to the set of content items and constraints as well as changes in the sign's context from the context and constraints parser, distributing the new set to the filter pipeline and initiating the lottery scheduler to generate a scheduling decision if necessary. The manager initiates a new scheduling iteration after each content item has been presented or if the context of constraints change.

Context and Constraints Parser

The input to the scheduling process is a set of content items and constraints. The context and constraints parser is capable of receiving and reading this set of content items and converting it into a format that can be read by filters and the lottery scheduler. This component enables the scheduling system to work with a range of signage systems that may use different formats for distributing schedules – helping to localise changes that may be necessary for different formats. Changes to the parser can be made without impacting other components of the scheduler.

Context Store

The context store is a common repository for storing contextual information about the sign. All components of the lottery scheduler (and the overall system) can feed information into the context store (e.g. sensors capturing information on passers-by). Other components, such as ticket allocators and filters, can request this information from the context store and utilise it to help inform more intelligent (context-aware) scheduling decisions.

Filter Pipeline

The filter pipeline consists of a series of independent filters that are able to process a set of content items and constraints and remove items whose constraints cannot be met at the present time. For example, constraints such as date/time can force an item to be only shown on certain times while a presence constraint might restrict the presentation of a content item to occurring only when a specific user is nearby. The filtering pipeline passes the set of content items through each filter in turn. The configuration of the filter pipeline and of each individual filter is stored in the context and configuration store.

Lottery Scheduler

The main component of the overall scheduling system is the lottery scheduler. The lottery scheduler receives a filtered set of eligible content items and distributes this set to all available ticket allocators. As described previously, there may be multiple scheduling requirements and each of these requirements can be represented by a separate ticket allocator. The modular design of the scheduler component allows new ticket allocators to be plugged in at any time in response to changing requirements.

The lottery manager distributes the same set of eligible content items and a fixed number of lottery tickets to each ticket allocator. Each ticket allocator then allocates lottery tickets to content items, and drops tickets into the ticket pool. The number of lottery tickets allocated is limited to the set of empty tickets distributed by the lottery manager. To enable a ticket allocation based on the context of the display (e.g. for personalised content) all ticket allocators are given access to the context and configuration store. The lottery manager will typically wait for all ticket allocators to be complete prior to making a scheduling decision, but ticket allocators can be interrupted by the manager if the ticket allocation process is taking too long to complete. All tickets dropped into the ticket pool at the point of the lottery are considered for the draw that determines a scheduling decision.

The scheduling manager decides when the draw should take place. This may be when all ticket allocators are ready, or, for example, when the ticket allocators have exceeded the maximum time for allocating tickets. Once the decision has been made to initiate a draw the scheduling manager will draw a random ticket from the pool. The description of the winning content item will be returned to the scheduler manager and sent to the display immediately.

In common with the filtering component, the scheduler manager and all ticket allocators are given access to the context information of the sign (described in more detail below). Context information can be used by ticket allocators to improve their allocation process. The scheduler manager can allow the user to specify which ticket allocator should be used, and how many empty tickets each of the ticket allocators should receive. This provides a mechanism for changing the content displayed and influencing the scheduling decision. Context-sensitive ticket allocators can also utilise this information from the context store – for example to reflect current viewer preferences.

Configuration
The configuration component is designed to enable display owners to configure the sign to their individual needs. In the context of the lottery scheduler, users can specify in the configuration which filters and ticket allocators should be used as well as how many tickets each allocator should distribute.

Implementation

We implemented a proof-of-concept lottery scheduler as a component within our existing e-Campus deployment at Lancaster [12]. Our deployment consists of approximately 30 Mac Minis running the Yarely software [4] and displaying content from a combination of e-Channels [12] and Mercury [5]. The e-Channels system focusses on organising content items into logical groups ('Channels') of items provided by content providers. Display owners subscribe their displays to content from one or more channels. The Mercury application store allows for the distribution of content modelled as applications, providing support for both traditional signage content and new types of interactive applications. Display owners can purchase applications from Mercury and add these to their displays. Both e-Channels and Mercury export display content subscriptions (content plus constraints) in the XML-based Content Descriptor Set (CDS) format described in Clinch et al. [4].

The lottery scheduler was implemented in Python (1,051 lines of code) as a plugin to the existing Yarely signage player running on our display nodes. Yarely accepts a CDS as input and controls both scheduling and playback. The Yarely software is a component-based system allowing easy substitution of subsystems and the lottery scheduling component proposed in this paper was therefore implemented as a direct replacement for Yarely's existing "Playlist Generation and Scheduling" component [4].

Each component of the lottery scheduler (illustrated in Figure 1) was built as a separate Python module. Within the filtering and lottery components, filters and ticket allocators were written as separate Python classes that can be registered within the system and communicate using ØMQ. Conceptually, lottery tickets are allocated to content items. However, in the underlying implementation each ticket allocator receives a set of pre-generated empty ticket instances that can each hold a reference to one content item. The ticket allocation process is threaded: ticket allocators run in parallel and are managed and monitored by the lottery manager.

EVALUATION

Benchmarking

We benchmarked our scheduler to explore its performance with varying numbers of content items and lottery tickets. This evaluation focussed on the time taken to make a scheduling decision.

Our benchmarks were performed on a Mac Mini (2.6 GHz Intel "Core i5", 8 GB 1600 MHz LPDDR3 SDRAM; 1 TB HDD). The Mac Mini ran Mac OS X 10.10.2 (Yosemite) and the Yarely digital signage player [4], our scheduling system was dropped in as a direct replacement to the existing scheduling and playlist generation component in Yarely. We benchmarked the system with two ticket pool sizes: $1,000$ and $10,000$, and a varying number of content items (all image files with resolution $1,000 \times 1,000$ and approximate file size of 0.6 MB) from 1–$10,000$. Each combination was measured 30 times. The results of our benchmarks are shown in Figure 2.

Overall we find that an increased number of tickets increases the time taken to schedule content. This is primarily due to an increase in the time taken to perform the ticket allocation – on average, the duration of the lottery increases 7.77 times with the ten-fold increase in tickets (1,000 tickets vs. 10,000). We attribute this increase in time taken for larger ticket pools to overhead in our implementation of the ticket allocation process—our use of Python Queue objects in the current implementation means that each allocation of a ticket typically results in a context switch between threads. Filtering based on constraints takes place prior to ticket allocation and hence the performance of this component is independent of the number of tickets allocated.

We find that an increased number of content items results in an increase in the time taken to schedule content. Running the lottery with 1,000 tickets takes 181.60, 736.0, 1477.47 milliseconds with 100, 1,000, and 10,000 content items respectively. This relationship is broadly linear [Figure 2], however we note that at the point at which the number of content items exceeds the number of tickets to be allocated the rate of increase in allocation time slows. Running the lottery with 10,000 tickets the time taken continues to increase steadily throughout —the draw takes 1317.73, 5962.23 and 18313.47 milliseconds for 100, 1,000, and 10,000 content items respectively. In this set of test cases we do not reach the point at which the number of content items exceeds the number of tickets and hence we do not see the change in behaviour witnessed with small numbers of tickets.

Unsurprisingly, the performance of other scheduling and Yarely components is unaffected by the number of tickets but does increase with number of content items. For example, filtering of content items takes 171.47, 950.57 and 10405.70 milliseconds for 100, 1,000, and 10,000 content items respectively. Overall time for Yarely to display images of this size on the Mac OS platform has previously been reported as approximately 1.5 seconds [3] (this includes a 0.6 second animation to fade items onto the screen).

In terms of overall performance we note that for smaller numbers of content items the item scheduling can be completed quickly (for example, the overall scheduling time for 100 items is less than a third of a second with 1,000 tickets). However, larger content sets do take considerably longer to schedule (overall scheduling time for 1,000 items is 1655.17 milliseconds and for 10,000 items 11581.73 milliseconds – with 1,000 tickets each). In practice, our experiences with a long-term display network indicate that displays are typically subscribed to $\sim$30 content items (mean 31.33, median 28.5, max:107) during any given period. Equally, the performance cost associated with very large content sets can of course be accommodated by the scheduler by simply starting the filter-

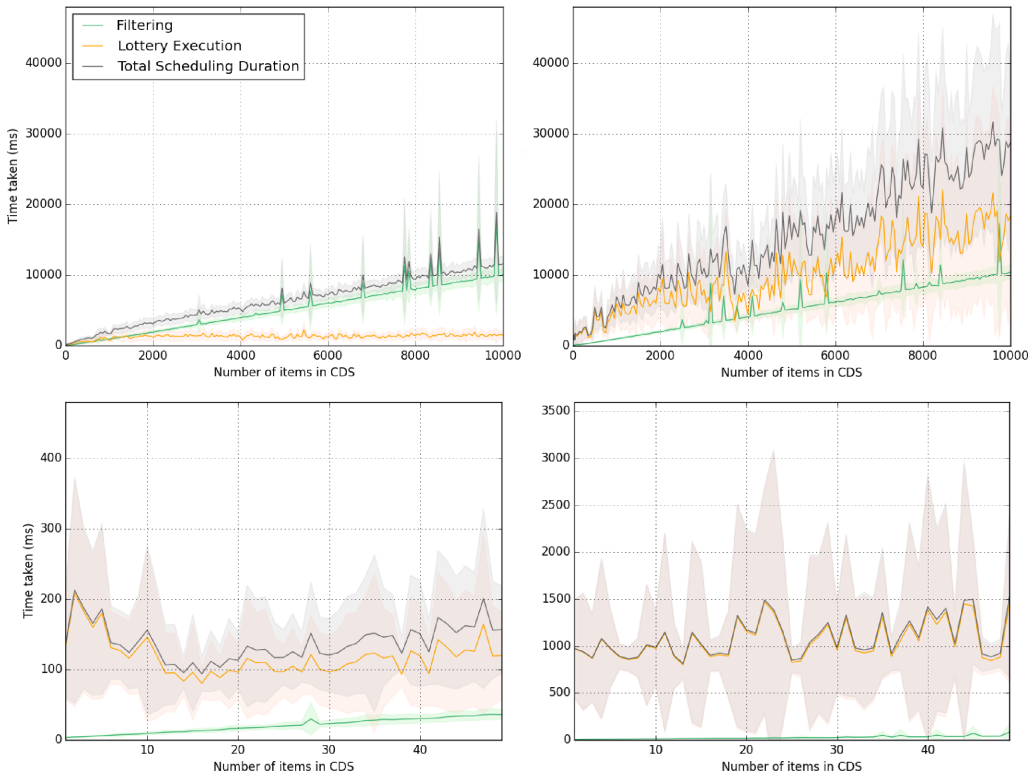

Figure 2: Time taken (mean and standard deviation) to schedule varying numbers of content items (top: 1–10,000 items, bottom: 1-50 items) using a lottery draw based on 1,000 tickets (left) and 10,000 tickets (right). The top plots show scalability, whilst the bottom plots provide indicative performance for real-world settings. All plots use the same legend (shown top left).

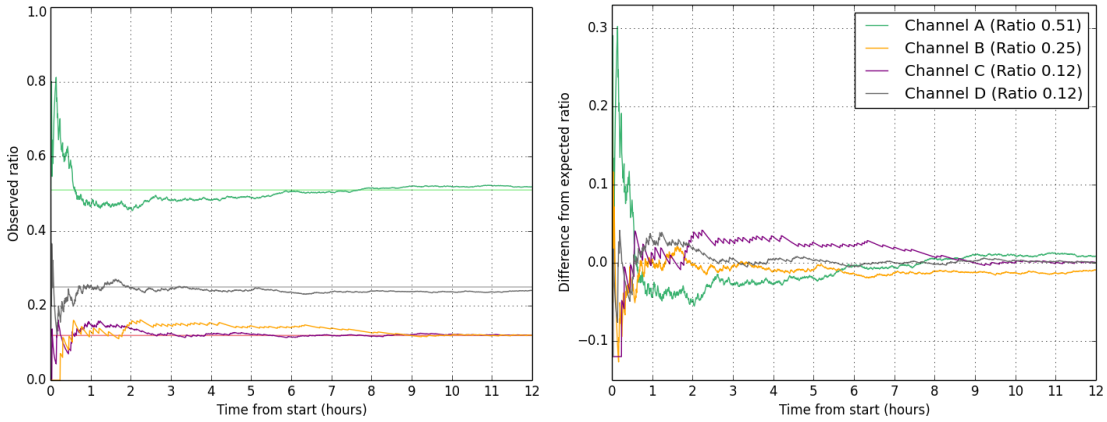

Figure 3: Accuracy of the lottery scheduler using a ratio allocator and sample e-Channel content. In the leftmost plot, observed ratios over 12 hours. In the rightmost, the difference between observed and expected ratios over the same 12 hours. Both plots use the same legend (shown right); the leftmost plot also includes paler lines to mark the expected ratio for each channel.

ing and lottery process before the current item reaches the end of its playback.

Real World Dataset

In addition to the above benchmarking, we wanted to understand the accuracy of our scheduler implementation in a real-world deployment. Our own campus deployment features content grouped into 'Channels' [12], with most displays drawing content from multiple channels and each channel allocated a specific ratio. We created a representative content set with four Channels that incorporated a range of content items from those currently playing on our deployment. Channel A contained 16 images and was allocated 0.51 of the airtime. Channel B contained four images and was allocated 0.25 of the airtime. Channel C contained a single 60-second video and was allocated 0.12 of the airtime. Channel D contained four images and was also allocated 0.12 of the airtime.

We ran the lottery scheduler with a single ticket allocator (the ratio-based allocator) and 10,000 tickets. The scheduler ran on a single display, scheduling and rendering content for 12 hours continuously, in line with the channel subscriptions described above. The video was played for its full duration, and all other content types for a default period of 15 seconds (i.e. total content duration across the channels was 420 seconds).

Our scheduler performed well in the deployment. The system quickly approximates the specified ratios [Figure 3] of $0.51 : 0.25 : 0.12 : 0.12$—during the first hour, the observed ratios move from $0.65 : 0.16 : 0.07 : 0.12$ (fifteen minutes in) to $0.59 : 0.22 : 0.11 : 0.08$ (half an hour in) and then to $0.50 : 0.25 : 0.13 : 0.13$ (forty-five minutes in). Although some variation is seen over time, after twelve hours of playback, the observed ratios are very close to the expected ratios at $0.52 : 0.24 : 0.12 : 0.12$.

Within the channels, all content items can be seen to be played fairly frequently. Our ticket allocator treated each item within the channel to an equal portion of the channel's ratio (taking into account the total duration of each item) and so we would expect to see items within a channel played with broadly the same share of the channel's overall screen time; this is roughly the case although we also see the effect of the randomness that lottery scheduling provides. Using Channel D as a case study, for example, we see that after the first hour, the channel had occupied the screen for just over nine minutes. Each of the four items in the channel had played for between 1.16 and 3.23 minutes each accounting for 12.86–35.77% of the channel's screen time (a completely even distribution would have given each 2.25 minutes, i.e. 25%). After four hours, Channel D had played for almost thirty minutes with each item receiving between 6.00 and 8.91 minutes (20.04–29.79% of the channel's screen time), and at the end our twelve hour study period, Channel D had played for almost 87 minutes with each item receiving between 16.16 and 25.70 minutes (18.64–29.65% of the channel's screen time).

Finally, we note that our scheduler demonstrates some behaviour not typically seen in other approaches (e.g. round robin). Due to the random selection of lottery tickets the same item may be presented multiple times consecutively. During our 12 hour run, we see items being played up to three times consecutively (i.e. an image that stays on the screen for 45 seconds rather than 15). Combining ratio-based selection with other ticket allocators (e.g. to prefer content that has not been played recently) may help to overcome this behaviour. Of course, our study only shows the effects of using a single ticket allocator. In practice we anticipate that combining multiple ticket allocators will result in variation in the accuracy of the ratios as the system attempts to balance a variety of ticket allocator preferences against each other.

RELATED WORK

Early pervasive display research systems provided little in the way of scheduling control—some were restricted to single items of content (e.g. media links), whilst others scheduled items only in response to specific presence or interaction events (e.g. [7]) or by simply 'cycling' through content items (i.e. round robin scheduling) (e.g. [13]). Combining interaction-driven and round-robin scheduling has continued as a common pattern for pervasive display systems (e.g. [11, 14, 2, 10]).

As digital signage has become more commonplace, a greater need for more complex scheduling behaviour has emerged. Payne et al. [15] proposed the use of auctions to select advertisements on public displays. Their display used Bluetooth scanning to count users in front of the display and build up user histories. A repetitive second-price sealed bid auction determined the winner from a set of advertising agents, each bidding on behalf of an advertisement to be shown.

Other approaches have attempted to combine multiple distinct algorithms in order to accommodate groups of scheduling preferences. For example, CommunityWall [16] used combinations of rules to generate priorities for available content—for example, to prefer items based on time of day, or those that had led users to interact. In commercial signage systems, a variety of timeline and constraints-based systems have emerged (e.g. Sony Ziris [17], BroadSign [1]). Such systems typically feature sophisticated user interfaces that offer display owners very fine-grained control over their displays but may be overly cumbersome for non-expert users [12].

An alternative approach for flexible accommodation of multiple scheduling concerns was conceived by Storz et al. [18] and Elhart et al. [10]. Both proposed providing APIs that would allow the implementation of complex schedulers to manage content transitions on a display node. Elhart et al. also identified a wide range of scheduling constraints and preferences and a notation that would allow description of a specific scheduling problem (i.e. a specific combination of scheduling and application environment together with the desired scheduling behaviour).

CONCLUSIONS

Despite significant research activity content, scheduling remains a key challenge for emerging pervasive display systems. We believe that this is is the first paper to explore the development of a generalised scheduling architecture for public displays based on lottery scheduling. Our initial implementation experiences and evaluation suggest that lottery scheduling, when combined with effective constraint processing, can form part of a comprehensive scheduling solution for pervasive displays. Our future work is to demonstrate the integration of our lottery scheduler with Tacita [6] to support personalisation and to explore whether lottery scheduling can be used to provide an appropriate metaphor for users when specifying scheduling preferences.

ACKNOWLEDGMENTS

This research is partially funded through the Future and Emerging Technologies (FET) programme within the 7th Framework Programme for Research of the European Commission, under FET grant number: 612933 (RECALL), and was made possible with the support of a Google Faculty Research Award and a Google Cloud Credits Award.

REFERENCES

1. BroadSign Digital Signage Software Solutions. `http://broadsign.com/` [Last accessed: February 2015].
2. Churchill, E. F., Nelson, L., Denoue, L., and Girgensohn, A. The plasma poster network: Posting multimedia content in public places. In *Proceedings of INTERACT '03*, IOS Press (2003).
3. Clinch, S. *Supporting User Appropriation of Public Displays*. PhD thesis, Lancaster University, 2013.
4. Clinch, S., Davies, N., Friday, A., and Clinch, G. Yarely – a software player for open pervasive display networks. In *Proceedings of PerDis '13*, ACM (2013).
5. Clinch, S., Mikusz, M., Greis, M., Davies, N., and Friday, A. Mercury: An application store for open display networks. In *Proceedings of UbiComp '14*, ACM (2014), 511–522.
6. Davies, N., Clinch, S., and Alt, F. *Pervasive Displays: Understanding the Future of Digital Signage*. Synthesis Lectures on Computer Science. Morgan & Claypool Publishers, 2014.
7. Davies, N., Friday, A., Newman, P., Rutlidge, S., and Storz, O. Using bluetooth device names to support interaction in smart environments. In *Proceedings of MobiSys '09* (2009).
8. Davies, N., Langheinrich, M., Clinch, S., Friday, A., Elhart, I., Kubitza, T., and Surajbali, B. Personalisation and privacy in future pervasive display networks. In *Proceedings of CHI '14*, ACM (2014).
9. Elhart, I., Langheinrich, M., Davies, N., and José, R. Key challenges in application and content scheduling for open pervasive display networks. In *Adjunct Proceedings of PerCom'13 : Works-in-Progress* (2013).
10. Elhart, I., Langheinrich, M., Memarovic, N., and Heikkinen, T. Scheduling interactive and concurrently running applications in pervasive display networks. In *Proceedings of PerDis '14*, ACM (2014).
11. Finney, J., Wade, S., Davies, N., and Friday, A. Flump: The FLexible Ubiquitous Monitor Project. In *Cabernet Radicals Workshop* (May 1996).
12. Friday, A., Davies, N., and Efstratiou, C. Reflections on long-term experiments with public displays. *Computer, IEEE 45*, 5 (May 2012), 34–41.
13. Kray, C., Galani, A., and Rohs, M. Facilitating opportunistic interaction with ambient displays. In *Workshop on Designing and Evaluating Mobile Phone-Based Interaction with Public Displays at CHI 2008* (2008).
14. McCarthy, J. F., Costa, T. J., and Liongosari, E. S. Unicast, outcast & groupcast: Three steps toward ubiquitous, peripheral displays. In *Proceedings of Ubicomp '01*, Springer-Verlag (2001), 332–345.
15. Payne, T., David, E., Jennings, N. R., and Sharifi, M. Auction mechanisms for efficient advertisement selection on public displays. In *Proceedings of ECAI 2006*, IOS Press (2006), 285–289.
16. Snowdon, D., and Grasso, A. Diffusing information in organizational settings: Learning from experience. In *Proceedings of CHI '02*, ACM (2002), 331–338.
17. Sony Ziris™. `http://pro.sony.com/bbsc/ssr/cat-digitalsignage/resource.solutions.bbsccms-assets-cat-digsignagedev-solutions-Ziris.shtml` [Last accessed: April 2014].
18. Storz, O., Friday, A., and Davies, N. Supporting content scheduling on situated public displays. *Computers & Graphics 30*, 5 (2006), 681–691.
19. Waldspurger, C. A., and Weihl, W. E. Lottery scheduling: Flexible proportional-share resource management. In *Proceedings of USENIX OSDI '94*, USENIX Association (1994).