SCORE: Exploiting Global Broadcasts to Create Offline Personal Channels for On-Demand Access

Gianfranco Nencioni, Nishanth Sastry, *Member, IEEE*,
Gareth Tyson, Vijay Badrinarayanan, Dmytro Karamshuk, *Member, IEEE*,
Jigna Chandaria, and Jon Crowcroft, *Fellow, IEEE, ACM*

***Abstract*—The last 5 years have seen a dramatic shift in media distribution. For decades, TV and radio were solely provisioned using push-based broadcast technologies, forcing people to adhere to fixed schedules. The introduction of catch-up services, however, has now augmented such delivery with online pull-based alternatives. Typically, these allow users to fetch content for a limited period after initial broadcast, allowing users flexibility in accessing content. Whereas previous work has investigated both of these technologies, this paper explores and contrasts them, focusing on the network consequences of moving towards this multifaceted delivery model. Using traces from nearly 6 million users of BBC iPlayer, one of the largest catch-up TV services, we study this shift from push- to pull-based access. We propose a novel technique for unifying both push- and pull-based delivery: the Speculative Content Offloading and Recording Engine (SCORE). SCORE operates as a set-top box, which interacts with both broadcast push and online pull services. Whenever users wish to access media, it automatically switches between these distribution mechanisms in an attempt to optimize energy efficiency and network resource utilization. SCORE also can predict user viewing patterns, automatically recording certain shows from the broadcast interface. Evaluations using our BBC iPlayer traces show that, based on parameter settings, an oracle with complete knowledge of user consumption can save nearly 77% of the energy, and over 90% of the peak bandwidth, of pure IP streaming. Optimizing for energy consumption, SCORE can recover nearly half of both traffic and energy savings.**

***Index Terms*—Content distribution networks, digital TV, digital video broadcasting, energy conservation, energy efficiency, environmental factors, machine intelligence, recommender systems, TV broadcasting, TV receivers.**

Manuscript received February 27, 2014; revised November 16, 2014; accepted June 15, 2015; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor S. Sen. This work was supported by the UK EPSRC under Projects No. EP/K024914/1 and EP/H040536/1, H2020-ICT-2014-2 projects 5G NORMA and VirtuWind, and the EU-INDIA project REACH. This paper is an extended version of a paper presented at the 22nd International World Wide Web Conference, Rio De Janeiro, Brazil, 2013.

G. Nencioni was with the University of Pisa, 56122 Pisa, Italy. He is now with the Norwegian University of Science and Technology, 7491 Trondheim, Norway (e-mail: g.nencioni@iet.unipi.it; g.nencioni@iet.unipi.it).

N. Sastry and D. Karamshuk are with King's College London, London WC2R 2LS, U.K. (e-mail: nishanth.sastry@kcl.ac.uk; dmytro.karamshuk@kcl.ac.uk).

G. Tyson is with Queen Mary University of London, London E1 4NS, U.K. (e-mail: gareth.tyson@eecs.qmul.ac.uk).

V. Badrinarayanan and J. Crowcroft are with the University of Cambridge, Cambridge CB2 1TN, U.K. (e-mail: vb292@cam.ac.uk; jon.crowcroft@cl.cam.ac.uk).

J. Chandaria is with BBC R&D, London W12 7SB, U.K. (e-mail: jigna@rd.bbc.co.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2015.2456186

I. INTRODUCTION

THE LAST 5 years have seen a dramatic shift in the way people interact with media services. Traditionally, those wishing to enjoy TV and radio shows were forced to watch them at prespecified broadcast times. Recently, however, broadcasters have begun to also make their content available online using on-demand services. This type of service is termed a "*catch-up*" system, allowing viewers to watch recently broadcast media for a specific period after its initial broadcast. This highlights a key shift in the way users consume TV content, moving from the traditional push model to a far more user-centric pull model. Perhaps the most prominent example of this is the BBC iPlayer, which allows users in the United Kingdom (UK) to pull nearly all of BBC's TV and radio shows from the Internet for (typically) 7 days after their initial broadcast. Launched at the end of 2007, the service has since exploded in popularity with an estimated 40% of UK households using it [30]. Although broadcast figures remain orders of magnitude more than corresponding iPlayer audiences, it is undeniable that catch-up has radically altered the way in which users access the BBC's content.

As more and more users start to rely on the flexibility of catch-up TV and move away from traditional TV broadcasts, it raises important questions about how to provision infrastructure for future TV audiences. For instance, by 2011, BBC iPlayer had become one of the largest applications by traffic volume on the UK Internet, second only to YouTube [31]. This has implications for network capacity provisioning: Traditional TV has managed to scale up to large audiences because of its reliance on broadcast infrastructure, but the costs of catch-up viewing increases with each stream. Additionally, this move towards individual, personalized online streaming is significantly increasing the collective energy consumption of TV content distribution: The BBC estimates that for all of its channels except one,[1] Digital Terrestrial Television (i.e., broadcast TV) has a smaller per-viewer carbon footprint than catch-up streaming. This is because broadcast has fixed carbon costs that can be amortized over large audience sizes, whereas the carbon costs of streaming grows with each additional user [12]. Motivated by these observations, we ask whether the flexibility of on-demand viewing can be supported while still relying as much as possible on low-energy broadcast.

With this in mind, we first explore how "catch-up" has changed TV viewing, using BBC iPlayer, the UK's largest TV

[1]The BBC Parliament channel, which has fewer viewers compared to other channels, is the sole exception.

and radio catch-up service, as a case-study. Using historical data of approximately 6 million users accessing radio and TV content on iPlayer, we seek to explore the key consequences of supplementing push-based broadcast delivery with a pull-based online equivalent. We find that many users choose to exploit the flexibility of online-pull, forming their own personalized bundles of preferred content and watching it in patterns specific to pull-based architectures (e.g., viewing multiple episodes of a TV series in a short timespan). That said, we also continue to observe push-like behavior such as viewing as soon as content is available and a general preference for newly released content. We also see evidence of high engagement, with high video completion ratios, and users consistently watching many episodes of favorite TV serials.

Through the above exploration, we highlight the unique benefits and potential of both traditional broadcast and online pull models. Using the access patterns we find, we design the Speculative Content Offloading and Recording Engine (SCORE) to combine the benefits of broadcast-based and pull-based access and reduce the cost of content delivery (both in terms of energy and network costs). Since our trace-driven study shows that users on catch-up are constructing highly personalized schedules of content to watch at their convenience, SCORE attempts to emulate this by predicting which shows a user is likely to watch, and then constructing personalized lists of favorite shows for each user. Episodes of favorite shows are then speculatively recorded on user-local storage such as digital video recorders (DVRs, also known as personal video recorders or PVRs), enabling later offline on-demand access. This process can remove significant amounts of energy-intensive IP traffic. Entire shows are recorded since the traces show relatively low rates of abandonment.

Thus, SCORE effectively embeds a personalized local catch-up service within DVRs and thereby offloads content from the Internet and from the over-the-top (OTT) catch-up TV service. When a show that has not been recorded is requested, it falls back to the current online pull-based model and streams the content item on-demand. Through this predictive offloading of iPlayer load, SCORE can mitigate the *network* footprint of catch-up services. Interestingly, recording on DVRs complying with EU regulations on power consumption of set-top boxes [1] can also decrease the nationwide *energy* footprint, compared to streaming.

The basic SCORE concept is pluggable and can be configured for optimizing either energy or traffic savings, given the amount of locally available storage as a constraint. We focus on energy savings for two reasons. First, sustainability is a major concern for public service broadcasters like the BBC [8]. Second, whereas it is clear that speculative recording of DTT broadcasts results in a nonnegative decrease in network traffic (with savings strictly positive when the user accesses the recorded item from local storage rather than via OTT catch-up), it is not *a priori* clear that energy can be saved because speculative recording incurs an upfront energy expense that only pays off if the recorded item is accessed by the user. To demonstrate this potential, we explicitly develop the optimization problem for saving energy by adding a penalty for the energy expense of recording, and evaluate the benefits. Note that the two benefits are not mutually exclusive—saving energy saves traffic, and the reverse could hold as well.

Our evaluations show that given access to just 32 GB of storage, an oracle with complete knowledge of users' future accesses and optimizing for net energy savings could, depending on parameter values of the energy model we use, the bit rate used for streaming, etc., save up to 97% of peak traffic, and up to 74% of the energy. For similar parameter values, the energy-optimizing version of SCORE is able to recover more than 60% of the energy and traffic savings obtained by the oracle. Dependency on parameter values is resolved using sensitivity analysis. Optimizing for traffic reductions rather than energy consumption, an additional 5%–15% traffic savings can be achieved (at the cost of energy).

SCORE can be incorporated as a software update into modern DVR architectures such as YouView. Considering that DVRs have over 50% penetration in major markets such as the US and UK [15], [29], and that common DVR standards including YouView allow over-the-air software updates [2], [36], we believe that deployment is highly feasible.

II. What is a Catch-Up Service?

Catch-up services offer temporary on-demand access to media that has been previously broadcast via traditional means (TV or radio). Its purpose, as the name suggests, is to allow users to "catch up" with shows that they have missed on broadcast. Within this paper, we focus on one prominent catch-up service, BBC iPlayer,[2] which we now detail.

A. BBC iPlayer

The BBC has a number of local and national TV and radio channels, which broadcast content over the air in the UK. The BBC makes this broadcast content freely available to UK viewers on the iPlayer Web site and apps for a fixed period of days after the broadcast, depending on content licensing terms and other policies. Thus, the iPlayer provides an alternate "over-the-top" access mechanism for content that is typically broadcast over the air. BBC iPlayer is widely used within the UK, by an estimated 40% of households [30]. This creates a significant infrastructural footprint, both in terms of energy and bandwidth consumption. BBC iPlayer streams are entirely free of advertisements since the content programming is supported by TV licensing fees. It is worth highlighting that, in contrast to traditional on-demand services, the content items on BBC iPlayer change constantly; new items are added (typically immediately after broadcast) and removed after a short timespan.

B. BBC iPlayer Dataset

This paper studies a dataset derived from 8 weeks of access logs to the BBC iPlayer catch-up service, from September 4 to October 31, 2010. One in every four accesses to iPlayer during this period is recorded in the access log, giving a 25% sample of all accesses. Each log entry contains a timestamp for the start and end of the stream for one content item to one user. Altogether, the trace consists of 32 691 343 streams from 5 985 458 users, accessing 37 728 unique content items (episodes) from 3518 programs broadcast over 73 channels.

[2]Sometimes shortened to iPlayer in the text.

In addition, the BBC maintains Web pages about each program and episode that has been broadcast. We have harvested this data to augment the historical access logs with additional information such as the genres of the content item, the time and channel of broadcast, and the theoretical duration of the content item.[3] We also identify each content item as belonging to one (or more) of 11 genre categories: kids, drama, learning, factual, music, news, religion and ethics (r&e), sport, weather, comedy, and entertainment (entert.). Each category has finer-grained subdivisions into genres.

III. Characteristics of On-Demand Access

The introduction of catch-up services such as iPlayer has introduced a whole new *pull-based* mechanism for on-demand consumption of TV and radio content traditionally pushed to users via broadcast. This section explores the benefits from the pull mechanism, and the extent to which users still follow push-like access patterns. We divide this study into two parts, first characterizing the content access preferences, and then the temporal access patterns.

A. Content Access Patterns

This section asks *what* items users watch when allowed flexibility to pull items on-demand. We consider three axes of choice: duration of content, the type or genre of content, and whether the item is serialized, i.e., whether it belongs to a TV series comprising several episodes in sequence.

In each case, we use the same method to determine user preferences. We first consider the distribution of the parameter (e.g., content duration, genre or serial/nonserial) in the content corpus. Next, we consider a weighted distribution of the same parameter, weighted by the number of accesses. Their relative proportions indicate user preferences: If a particular value of a parameter is overweighted in the weighted distribution compared to the content corpus, then users prefer that value. If underweighted, users dislike that value.

1) Users Prefer Serialized Content: We first inspect the preference users have for serialized content. We find that serial content constitutes roughly 53.3% of the content corpus. Yet, in the list of items watched, serial content constitutes nearly 79.5%. Thus, it is evident that serialized content is disproportionately popular. This is a curious attribute of catch-up TV, which, in contrast to other platforms that consist more prominently of "one-off" shows such as movies on Netflix, or the shorter clips often seen in user generated repositories such as YouTube, is often driven more prominently by well-known serials (e.g., soap operas, comedy serials). That said, it is interesting to note that nearly half of all the content corpus is nonserial, suggesting that the BBC does invest significant amounts of airtime to broadcasting such content. On closer inspection, we find that traditional nonserial content (e.g., documentaries) does constitute a large fraction of the corpus, but simply does not gain the popularity of other serial-oriented genres (e.g., comedy, drama). This is likely a combination of many factors, not least the long history the BBC has in producing widely appreciated serial shows. Communication theorists also believe that strict, predictable schedules of serialized shows establishes viewing habits that become automatic [17, p. 19].

[3]Access log duration may differ from theoretical duration if users stop viewing before completion, e.g., due to network issues or of their own volition.

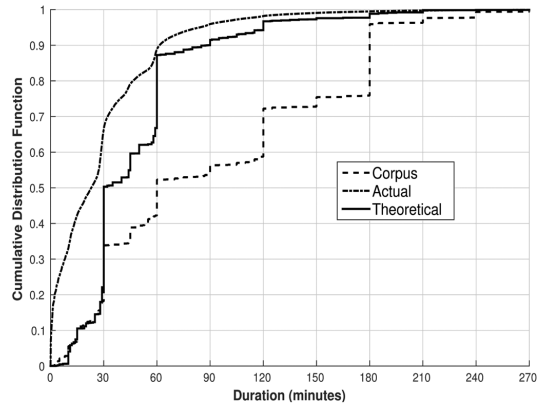


Fig. 1. Content length distributions: *Corpus* shows the distribution of durations for all items in the content corpus. *Theoretical* is the distribution of content lengths weighted by number of views. *Actual* shows the observed distribution of stream lengths. The content corpus has the most uniform distribution of content lengths. The theoretical distribution has nearly 90% of its mass under 60 min, showing that users prefer content shorter than an hour. Theoretical and actual distributions are close reconfirming low abandonment rates.

2) Users Prefer Short Duration Content: Fig. 1 considers three distributions of content durations, *corpus*, *theoretical*, and *actual*. *Corpus* is the distribution of content durations for each item in the catch-up content corpus. *Theoretical* is the distribution of durations obtained by weighting each item by the number of times it is accessed. *Corpus* is much more uniformly distributed than *theoretical*, which has most of its mass under 1 h. Furthermore, the relative mass of *theoretical* increases dramatically at two points: 30 and 60 min, which corresponds to standard durations of serialized TV shows. This indicates the relative popularity of these two kinds of content. The third distribution, *actual*, gives the actual durations of streams observed. The difference between *theoretical* and *actual* is an indication of how much of the content is actually watched. We note that only $\approx$25% of the requests are abandoned in the first 5 min, indicating that three quarters of users are engaged and watch a large proportion of the show. This is best highlighted by the close alignment between the *theoretical* and *actual* curves in Fig. 1.

3) Users Prefer Specific Genre Categories: Next, in Fig. 2, we consider the relative proportions of different genre categories in the content corpus compared to their proportions when weighted by the number of accesses. Categories where the watched bar is taller than the corpus are overweighted, and hence preferred by users. This clearly indicates a strong preference for certain categories such as drama, comedy, and kids' shows. In contrast, genre categories such as factual programs, music, and news constitute a large proportion of the content corpus but are not watched as much. Thus, although a public service broadcaster might provide a balanced content catalog, users tend to prefer common kinds of entertainment.

Given such strong preferences, we ask whether genres are a better way to create pull-based "channels" for users than the current broadcast channels. To answer this, we quantify how well a given partition of content items—into channels or genres—captures the content consumption history of individual users. Specifically, we compare the *self-information* [14] of describing users by the channels of their content items to that of describing users by genres of the items they consume. The higher the self-information is, the more information it captures of a user. Recall that the entropy of a random variable

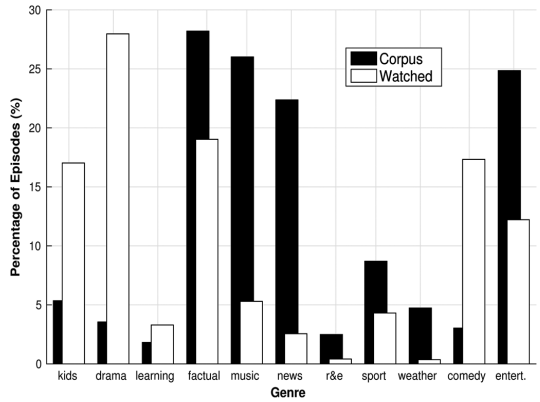


Fig. 2. Distribution of genre categories showing that drama, comedy, and kids' programs are overweighted w.r.t. corpus.

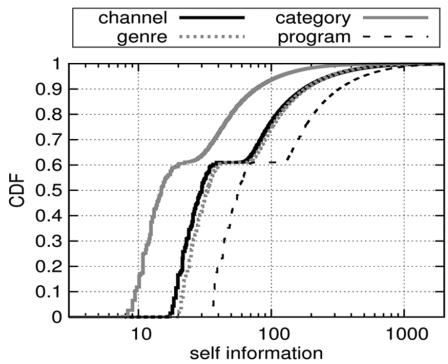


Fig. 3. Self-information of various content bundling strategies.

is obtained by taking the expectation of its self-information. The higher the entropy of a partitioning method, the better its representation of users is, on average, for the entire population.

Formally, let C be a set of content items available in the system and B be a *bundling* of content defined as a partition of C into N subsets (i.e., N bundles). Examples of bundling include partitioning the set of programs based on the channels they are broadcast on, or partitioning based on genres, with each channel or genre forming a bundle, respectively. For a given bundling B, we denote the watching history of a user with tuple $t_B = (n_1, n_2, \ldots, n_N)$, where n_j is the number of times a content item from a bundle $j \in B$ was watched by the user. Given a bundling method, we are interested in the self-information of the random variable T_B, $I(T_B) = -\log P(T_B = t_B)$. Note that $P(T_B = t_B)$ is given by the multinomial distribution

$$I(T_B) = -\log\left(\frac{l!}{n_1!n_2!\ldots n_N!} \times p_1^{n_1} p_2^{n_2} \ldots p_N^{n_N}\right) \quad (1)$$

where p_j is the probability of randomly choosing an item from bundle j, and l is the number of user's sessions, i.e., $l = \sum n_j$.

Fig. 3 plots this value for several bundling strategies: bundling programs into the current set of channels; bundling into one of the 11 coarse-grained genre categories; bundling into fine-grained genres; and, finally, bundling into individual programs, as an example of extremely fine-grained bundling. As expected, program-based bundling has the highest self-information. Interestingly, despite the population as a whole favoring certain genres over others, *channels defined for push-based broadcast capture users' consumption patterns better than genre categories*. However, when genre categories are split into finer-grained genres, user interests are captured with similar amount of self-information as broadcast channels.

B. Temporal Characteristics

A key feature of the pull model is that it creates temporal flexibility—users can choose when they consume content, rather than adhering to a push schedule. This leads to two benefits: At the infrastructure level, we see a flatter demand pattern as users are not restricted to the evening prime-time hours if they watch popular content. At the same time, users are able to consume content in a bursty fashion, for instance, watching multiple episodes in short time periods. Despite these trends, we also see access patterns that resemble push-like consumption, with a preference for fresh content, and spikes in access as soon as content is made available on the platform.

1) Pull Flattens Demand: To explore how viewers make use of the temporal flexibility of pull, Fig. 4 depicts the average number of requests received per hour across the whole trace. We plot two curves: The first (marked *broadcasting time*) plots access frequency by the original broadcast time of the content being requested; the second (marked *request time*) plots access frequency by the request timestamps in our traces. For example, suppose a primetime TV show was broadcast at 9 PM in the night but was requested at 10 AM the following morning. This request would be placed in the 10 AM bucket for the *request time* and 9 PM for the *broadcasting time*.

It can be seen that the access patterns of users in the pull model change significantly compared to broadcast. By allowing users to select when they consume content, requests are flattened far more over the day: When inspecting the broadcasting time, huge demand peaks occur for content broadcast between 18:00–20:00 for radio, and 19:00–23:00 for TV (corresponding to traditional "prime time"). In contrast, these peaks are flattened greatly in the request times of on-demand access. That said, it is evident that content that is broadcast during the peak time also dominates in catch-up service with greater volumes of access, indicating that broadcasters do an effective job of scheduling popular shows. The same (popular) items are watched in both pull and push models; albeit at different times.

Furthermore, the demand patterns are different between TV and radio content. Whereas TV has pronounced diurnal patterns with large numbers of requests during evening peak or prime time hours, radio has a flatter demand pattern, with its peak hours actually occurring during the afternoon. From an infrastructure perspective, these differences in peak times could be exploited by hosting both TV and radio content on the same delivery infrastructure, which can be used more efficiently throughout the day.

2) Pull Allows Bursty Access: Anecdotal evidence suggests that it is increasingly popular for people to spend evenings watching several episodes of particular shows. More generally, users can "catch up" on multiple episodes over time spans shorter than a week, the typical duration between consecutive episodes for serialized broadcast content. This is a key flexibility of the pull-based model in contrast with push-based delivery, where shows must be broadcast following predetermined schedules.

To quantify such bursty behavior, Fig. 5 presents a cumulative distribution function (CDF) of the number of episodes from the same TV show requested over various time periods by individual users. It can be seen that a small, but noticeable, number of users do exhibit burstiness when consuming media for both radio and TV, with slightly more multiple accesses in radio. For

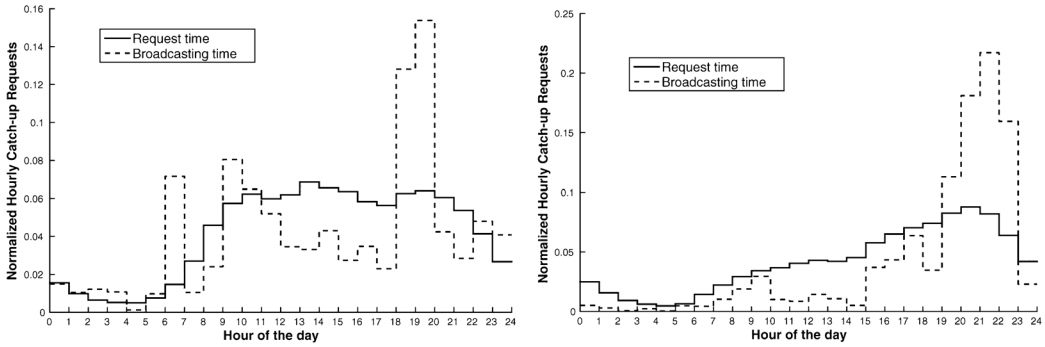


Fig. 4. Normalized distributions of catch-up request times by hour of day, and the broadcast times of requested items. The normalization is with respect to the daily number of requests (i.e., each data point is presented as a fraction of total daily viewing figures). Items broadcast during 7–11 PM "prime time" are very popular on catch-up, but request distribution is flatter. (a) Radio. (b) TV.

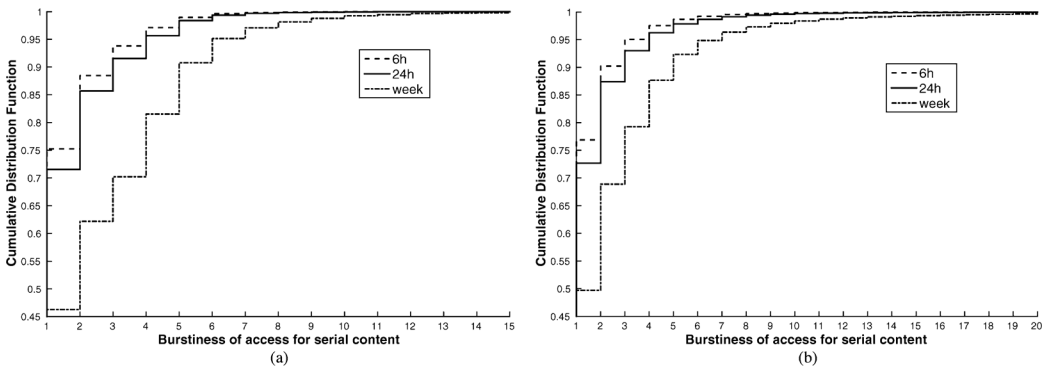


Fig. 5. Burstiness of accesses for serial content: CDF of the number of accesses from the same user for different episodes of the same serialized program within a time window (windows size: 6 h, 24 h, and 1 week) by considering users that have at least 10 logs in the whole dataset and programs that have at least four different episodes. Note that the full range of the y-axes for both figures is 0–1, but the figures are cut off at $y = 0.45$ to show the variation clearly. (a) Radio. (b) TV.

example, we find that ≈10% of the time, users watch multiple (> 1) TV episodes from the same program within a 6-h period, and nearly ≈30% do so within a week.

Two sets of factors of the current system might actually limit the extent of such bursty accesses. The first is the nature of the content. Some kinds of shows (e.g., news, weather) are outdated soon after release, or when a new episode is uploaded. Many programs in the UK tend to have fewer episodes than elsewhere (e.g., 6 episodes is common for a TV series in contrast to 13 or 26 episodes typical in other nations). This limits the maximum size of bursts. Additionally, iPlayer carries so called "long-form" content (e.g., TV episodes tend to be 60 or 30 min long), which limits the number of episodes that can be consumed over very short time periods.

The second set of limiting factors arise as a product of the way content is managed on iPlayer. Content is only available for catch-up if it has been broadcast previously. Similarly, content is periodically removed according to predetermined rules (driven by licensing and other policies), typically after the last episode of a show. Thus, during the early weeks of a serialized show, the size of bursts is limited by the number of episodes broadcast, whereas later on, typically after the final episode is broadcast, some early episodes may have expired.

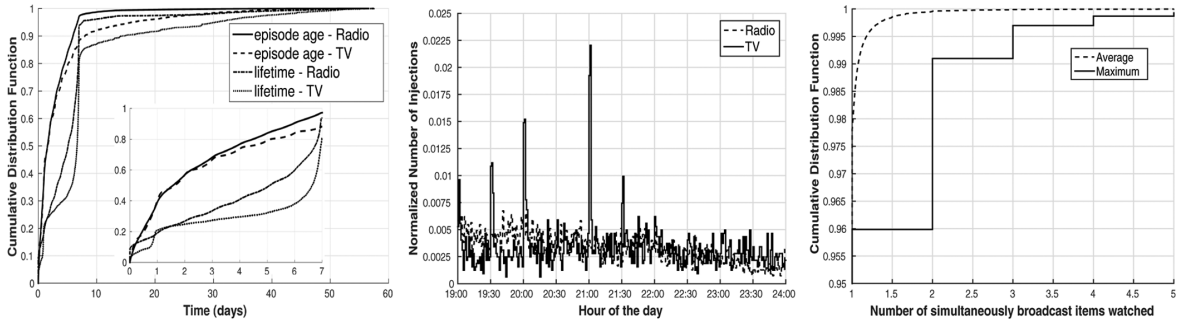
Regardless of these system limitations, some unique to the platform, some to the content corpus, there appears to be a non-trivial appetite for bursty consumption of multiple episodes of content over short periods of time, which is catered to by the pull model. Future system designs for on-demand access can better support such needs, for example, by creating content bundles comprising all episodes of a particular show.

3) Push-Like Access Patterns—Preference for Fresh Content: Although iPlayer allows for on-demand access, the limited availability of content on the platform, as well as the outdating of certain kinds of content such as news and weather, place limits on delayed viewing, as discussed in Section II.

To quantify this, Fig. 6(a) plots a CDF of the freshness of content, according to two metrics: *Lifetime* shows the length of time between the first and last view for each content item, and captures the rate at which content gets outdated. *Episode Age* shows the age of content items at each distinct view. It can be seen that there is a skew towards watching content soon after release. Almost 50% of views occur on the first day, even though much of the content does not get outdated until later on (average lifetime is ≈7 days). Over 90% of views happen within a week.

Notable differences also seem to appear between on-demand access for radio and TV. Fig. 6(a) shows that more radio content gets outdated early on: Whereas similar proportions of TV and radio content tend to get watched in the early stages of their release (e.g., under 4 days), TV viewers more slowly tail off as the content ages (after fourth day), as compared to radio, where over 95% of users listen to radio within the first 7 days of its release. This may be a product of radio's greater temporal dependency, where shows tend to relate to real-world events (e.g., topical discussions or talk shows).

Thus, it appears that users are broadly using catch-up for recent broadcasts, creating *a strong preference for fresh content, akin to push-based consumption*. We note that this preference for fresh content has been observed in other systems with progressive content releases [3]. However, our dataset also shows an interestingly strict adherence to broadcast schedule on the



(a) (b) (c)

Fig. 6. Push-like access patterns: (a) Preference for fresh content. Age of episodes at time of access versus lifetime of episode (time between last and first access), showing that most accesses happen early on, when content is still fresh. The inset graph zooms into the first week of accesses. (b) Adherence to schedule. Normalized number of first views in each time interval of 1 min between 7 PM–12 AM of every day, showing an adherence to broadcast schedule for eagerly awaited content (c) Serializibility of accesses. CDF of the number of contents simultaneously broadcast and watched by a user. Both the maximum (per user), and average values are shown. Over 96% have a maximum value of 1, and over 99.99% have an average of 1.1. Note that the y-axis range has been set to 0.95–1.

part of several users. Fig. 6(b) plots the number of first views that occur to each content on a minutely basis. For clarity, we focus on the evening peak hours, when the majority of requests are made (see Fig. 4) and also the maximum number of channels are broadcasting. It can be seen that especially with TV content, the first views spike strongly on the hour and half-hour marks, immediately after the content is put up on the platform, suggesting a strong push-like demand for accessing eagerly awaited content as soon as it is made available. Similar access patterns are seen outside the evening peak hours; although the spikes are strongest in the evening.

4) Push-Friendly Serializable Access Pattern: In the pull paradigm, if a user is interested in content being broadcast over two channels simultaneously, they can simply fetch it on-demand one after another, in a serialized fashion. Fig. 6(c) shows that despite this flexibility, users tend not to be interested in simultaneously broadcast content: Over 96% of users never need to watch content items that are broadcast simultaneously. On average, for over 99% of users, the average number of simultaneously broadcast shows that they are interested in is 1.1 or fewer. We conjecture that this is the result of careful planning of TV channel schedules to ensure that audiences interested in the same content items can watch them at broadcast time. Such planning is known to take into account not only the different channels of a single broadcaster such as BBC, but also the popular shows of competing broadcasters, to ensure maximum audience sizes. One implication of this is that if each user had personal "virtual channels" constructed by merging the different public broadcast channels, then one (or at most two) channels would suffice for nearly all users.

IV. SCORE: Offloading On-Demand Access

Section III has explored the characteristics of on-demand catch-up, showing that while it benefits from the pull model of on-demand access, it still needs to support push-like access patterns. With this in mind, we now propose a new system capable of exploiting these observations: the Speculative Content Offloading and Recording Engine. SCORE connects to both broadcast services and the Internet, unifying access to these mediums from the viewer's perspective via a set-top box. Whenever a user wishes to consume content, SCORE transparently decides how best to access it: via broadcast (if at the appropriate time) or via online pull (if it is later on). Importantly, SCORE also *integrates the principles of these two models* by intelligently recording popular content from the broadcast interface, creating local personalized bundles for individual users, by predicting their viewing patterns. This has clear benefits for users by providing an extremely high-performance local catch-up service that is not limited by network capacity and performance. However, the benefits extend beyond this. Specifically, we identify the potential to significantly decrease the energy footprint of content delivery by offloading traffic from the costly IP network onto the broadcast network instead (via automated recording).[4]

A. Designing SCORE

We start by considering the implications of the trace-driven measurements of Section III for the design of SCORE and derive the following design choices and simplifications.

1) Speculative Recording for On-Demand Access: The support for time-shifted viewing is used extensively: Fig. 4 shows that although content broadcast during TV prime time is also popular on catch-up and has the largest audiences, audience accesses for catch-up TV are more distributed in time. On the one hand, this decreases the overall load of simultaneous unicast streams to the server, leading to better network utilization. On the other hand, on-demand access also renders it difficult to share resources using multiuser reception mechanisms such as multicast, which would be ideal for amortizing costs across large audiences. In designing SCORE, these considerations lead us to derive amortized cost savings by exploiting an alternate broadcast channel available to BBC programs: Digital Terrestrial Transmission (DTT). We offer on-demand access by *speculatively recording broadcasts of content items* predicted to be watched later.

2) Whole Item Recording: Users show a high engagement: In contrast with the previously reported high levels of short-intervalled viewing due to channel surfing[5] in traditional (live) TV [11], [37], the proportion of short-intervalled catch-up streams (i.e., streams abandoned or stopped after a short period of viewing) is relatively small (Fig. 1). This stronger commitment suggests a simplified speculative recording scheme that stores entire items rather than hedging bets by storing a

[4]The rest of this section discusses the use of SCORE with energy efficiency as the objective. However, this choice is pluggable; an alternative that optimizes for network traffic is explored in Section VI-B. We also focus on the use of SCORE for TV, but the principle is equally applicable to radio.

[5]Also called channel "zapping" or "scanning."

"sampler" such as the first few minutes of a content item. Our decision to store entire content items is also influenced by the relative energy costs of recording broadcasts and on-demand network streaming: As described later, DVR recording is generally greener than streaming; thus recording entire shows can deliver more savings than recording samples.

3) Program History-Based Prediction: Users exhibit strong personalized preferences (Sections III-A-1–III-A-3); thus speculative recording needs to be based on personalized predictions. In particular, users' affinity to watch many episodes of the same program has the highest self-information (Fig. 3), leading us to design simple personalized predictors based on program history. As expected, this leads to the best performance, but we also report the performance of alternative prediction mechanisms in Section VI.

4) Expiration-Based Content Replacement and Weekly Cache Refills: Fig. 6(a) shows a strong push-like preference for fresh content with nearly 90% of accesses being for content broadcast less than a week before. It also shows that over 80% of items expire within 7 days of broadcast and cannot be watched later even if the user so wishes. In addition, it is common for TV shows to follow a weekly cycle, with new episodes broadcast around the same time each week.

Driven by these observations, we adopt an extremely simple cache management policy for SCORE: SCORE is run on a weekly basis, and a schedule of new recordings for the rest of the week is decided based on previous watching history. We assume that amount of storage available for each week is constrained by a fixed amount S. This limit can be set by the user, or reasonable defaults can be set automatically depending on a variety of factors, such as the total storage available on the DVR, or the bit rate encoding used. Given a specific storage constraint S and an objective such as minimizing energy or traffic footprint, SCORE speculatively decides the best schedule of items to store based on the predicted probability of access. However, once an item has been recorded, we do not actively evict it from the cache, but allow it to be removed naturally when the content expires or once it has been watched by the user. Thus, content items can remain for longer than a week, but we expect the number of such items to be small given the nature of the content corpus.

B. Overview of Operation

Fig. 7 shows a schematic of the SCORE DVR. Content can be acquired either from the DTT interface during broadcast time, or pulled from the IP network interface. For each content item requested by a user, a coordinator decides whether to show the content from: 1) the DTT interface if the content is being broadcast live when the user requests to view; 2) the DVR if the content is locally stored; or 3) IP streaming from the catch-up servers, if not stored locally. This unified approach hides complexity from the user, automatically obtaining the content from the preferred means without intervention.

SCORE's key novelty comes in its ability to create personalized bundles by learning and predicting viewing preferences. Exploiting this, SCORE automatically records and stores items speculatively from the broadcast channel. The SCORE element consists of a *predictor* and an *optimizer*. The predictor calculates weighting factors for each content item based on the program series to which it belongs. The decision on which items will be recorded (from the broadcast channel) speculatively is made by an optimizer, which calculates the expected utility of speculatively recording an item, subject to the storage limitations, and the other items that are due to be broadcast. The SCORE optimizer is run at the beginning of every week, using the upcoming broadcast schedule and the user's previous catch-up viewing history as inputs. The output is a schedule of content items to record speculatively from the DTT interface. SCORE wakes up the DVR from sleep/stand by at the scheduled broadcast time, records the item, and goes back to sleep. This therefore allows the user to stream the content locally, rather than use pull-based delivery via the Internet.

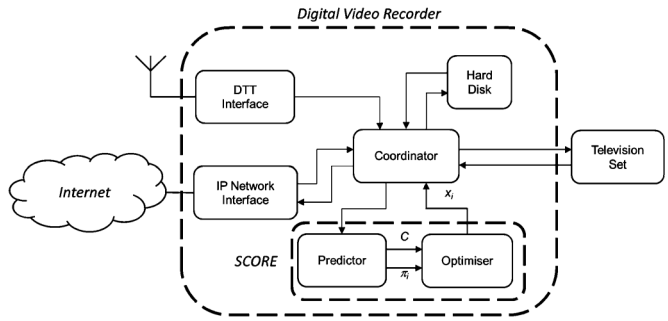


Fig. 7. Schematic of a DVR/STB with SCORE.

C. Optimizer

First, we describe SCORE's optimizer component. Speculative recording will never increase network traffic, but recording content not watched later on wastes energy. Although savings from watched items can compensate for unwatched items over a set of recordings, there can still be net energy loss. This is particularly undesirable, as these losses will be incurred by the viewer (in terms of their energy bills). As such, it is critical to ensure that energy reductions occur in a wider context, creating benefits across all stakeholders (both in the home *and* networking infrastructure). Consequently, we conservatively offload only content that is expected to minimize the overall energy spent in providing catch-up functionality.

Deciding which items to record can be formulated as a *binary integer linear programming problem*. Formally, given a set of content items C that are known to be broadcast in a given week, and a space constraint that a maximum of S bits can be stored, the task of the optimizer is to compute a binary valued variable $x_i \in \{0, 1\}$ for each item $i \in C$. $x_i = 1$ if i is stored in the DVR, 0 otherwise. The decision is based on P^{IP}, the power consumption characteristics of the IP streaming option, P^{DVR}, the power consumed by the DVR for speculative recording, and the characteristics of the content item: the duration τ_i and the bit rate encoding r, which determine the space occupied, and a weighting factor $\pi_{p_i} \in [0, 1]$ that encodes the probability that the user will watch item $i \in C$ based on the TV series p_i that i is part of.

We model energy consumed in the Internet by on-demand streaming in terms of an energy *per bit* figure E^b, following Baliga *et al.* [7]. This is a well-known and widely used model for capturing the energy consumption of a network infrastructure. Although it cannot provide exact measurements of energy consumption, it is built upon a realistic design of a countrywide network, assuming data from commercially deployed networking equipment. It also uses a nationwide video-on-demand service as a driving case study, therefore closely matching

our needs. As such, we find it an effective choice to use for SCORE, as even loosely accurate energy predictions allow SCORE to make effective decisions (as we later show). As with any such model, however, we are required to perform several approximations. Section V-A provides numerical details and discusses how we resolve the dependency on the E^b value by sensitivity analysis. In practice, for the storage levels we assume, the savings realized are relatively insensitive to E^b, especially for higher bit rates, which are indicative of future trends. Speculative recording on the DVR can therefore save energy only if

$$P^{\mathrm{IP}} = E^b * r > P^{\mathrm{DVR}}. \tag{2}$$

It is important to note that speculative recording cannot be used bluntly. It can waste energy in either of two ways. First, the optimizer might decide to store an item that is subsequently never watched; thus, wasting the energy involved in speculatively storing the item in the DVR. Second, the optimizer might decide not to store a content item that is subsequently streamed by the user, incurring a larger energy footprint than recording.

The function of the optimizer is therefore to minimize wasted energy expenditure while speculatively recording content. This is encoded in the following decision problem:

$$\text{minimize} \sum_{i \in C} \pi_{p_i} \cdot P^{\mathrm{IP}} \cdot (1 - x_i) + \sum_{i \in C} (1 - \pi_{p_i}) \cdot P^{\mathrm{DVR}} \cdot x_i \tag{3}$$

$$\text{subject to} \sum_{i \in C} r \cdot \tau_i \cdot x_i \leq S. \tag{4}$$

The objective function (3) is composed of two addends. The first computes the expected power spent for streaming items that the optimizer decides not to store, based on a probability of watching π_{p_i}. The second addend computes the expected power spent speculatively recording content that is not subsequently watched, based on the probability of not watching $1-\pi_{p_i}$. Equation (4) imposes the constraint that the amount of stored contents must to be smaller or equal to the size of the memory S available on the DVR.

Simplifications for Practical Application: In theory, solving the above decision problem accurately is a *0–1 Knapsack problem*, which is well known to be NP-hard. However, we can adopt a greedy approach and select content items one by one in descending order of the objective function value (3) until we run out of space S. This works well in practice because most high probability content items are 30- or 60-min programs; thus, this heuristic fills available storage except for a small slot usually $<$60 min long.

Similarly, in theory, it is possible that the resulting schedules generated by SCORE may contain more than two items that are broadcast simultaneously. Given that typical DVRs have two tuners, it is not feasible to record all simultaneous broadcasts. However, as described in Section III-B-4, users are in general interested in only one among the items that share the same airtime. For the rare cases when the recording schedule generated by SCORE may require simultaneously broadcast shows (this happens on average for 0.01% of users), it may be possible to exploit the fact that many shows have repeat broadcasts and record at a later time (assuming the user has not streamed from iPlayer before the repeat). Unfortunately, our dataset does not contain times of all subsequent repeats of a program, so we are unable to quantify (in Section V) the benefits of utilizing repeats for speculative recordings. In extremely rare cases, it may mean that some shows are not able to be recorded and need to be streamed. Equally, it is possible that the user has a more advanced DVR or simply has additional TV tuners installed to handle the case. Given that the vast majority of users do not watch simultaneously broadcast shows on catch-up, we consider this a corner case, and rather than complicate the optimization problem for all users, we handle the recordings as a "best effort": In case of conflict, SCORE could simply choose to record the content with the higher π_{p_i}.

D. Weighting Factors

To be usable in the optimizer, the end requirement from a weighting model M is a weighting factor $0 \leq \pi_p^M(u) \leq 1$ for each user u and program p, with larger π_p^M indicating greater confidence that episodes of p will be watched via IP streaming.

The episodic nature of TV programs and the strong preference of users for serialized content, as discovered in Section III-A-1, gives a simple but powerful *history-based* weighting model: Watching previous episodes of a series is a good indication that the future episodes will also be watched. Formally, a weighting factor π_p^H can be derived for a user u who has previously watched n_p^u episodes of a program p with n_p episodes, as the probability of watching that program

$$\pi_p^H(u) = \frac{n_p^u}{n_p}. \tag{5}$$

Plugging in $\pi_{p_i} = \pi_p^H(u)$ in the optimization problem (3)–(4) obtains the best performance among the alternatives we have tried. Therefore, our main evaluation of SCORE uses this weighting factor. This holds for the content makeup on BBC iPlayer, however this is not generalizable to all content repositories. As such, alternative models would be required for different repository types (e.g., movies); other weighting factors are explored in Section VI.

V. Performance Analysis

This section analyzes the performance of SCORE using the trace discussed before (Section II-B). We compute the aggregate energy and traffic savings achieved when SCORE is run by users in our trace and present the results as percentage savings. We first discuss the simulation parameters used (Section V-A). Then, we assess the energy (Section V-B) and traffic (Section V-C) savings achieved by SCORE. In each case, we first use an oracle-based approach to compute the theoretical limits of the savings achievable by speculative recording. Next, the savings achieved by SCORE is measured relative to the oracle. The dependence on parameter values is resolved by sensitivity analysis across the range of possible values for all parameter combinations.

In computing the list of content items to speculatively record, we focus on weeks 4–6 of our 8-week trace. This allows SCORE to work with the previous 3 weeks of history for the predictor, and at least 2 weeks after the broadcast for the user to watch the show, allowing a better estimation of achievable savings.

A. Parameters for Trace-Driven Simulation

SCORE balances two factors that contribute to energy consumption other than on the content provider servers. The first factor is the energy consumed on DVRs to record the content. We conservatively consider HD double-tuner DVRs, which are the most energy-intensive of the simple set-top boxes under EU

regulations. EU regulations [1] mandate a maximum power consumption of 13 W when turned on or on active standby, and 1 W when on passive standby. DVRs must also automatically be switched into standby mode when not in use. The SCORE DVR must therefore adhere to these requirements. Hence, the power consumption *added* by speculatively storing a content in the DVR, P^{DVR}, is conservatively taken as the maximum power difference possible between on and stand by states, i.e., 12 W. For the experiments, we assume that users do not use their DVRs, as this represents the worst-case scenario for SCORE (i.e., it is necessary to take the DVR out of standby for *all* speculative recordings).

The second factor, the energy spent in the IP network to transport the content to the user, is much harder to quantify. However, this is vital to measure the combined energy impact of both the network infrastructure and the home environment. Our use case of distributing content from a national broadcaster to audiences within the country over the public Internet closely fits the assumed model of Baliga *et al.* [7], which is based on a paper design of a national-level network in a broadband-enabled country, and includes a video distribution network for applications such as Video on Demand. The model makes detailed calculations using realistic numbers from various networking equipment currently deployed commercially. It therefore provides an effective and convenient method to calculate energy consumption parameterized in terms of E^b, the average energy per bit transported. However, as with other current energy models for the Internet, this introduces assumptions about the models and technology of networking equipment used, network hops from server to user, network over-provisioning and multiplexing levels, etc. To account for these uncertainties, Baliga *et al.* derive a range of values possible for this figure, from $E^b = 75$ μJ for current networks down to $E^b = 2$ μJ, for a future energy-efficient all-optical network. Power consumed can be calculated as $P^{\mathrm{IP}} = E^b r$, where r is the bit rate encoding of the content provider. Given the inherent uncertainty and approximations involved in coming up with these values, we perform a sensitivity analysis over a wide range of values. This allows us to model the energy use for a large set of potential networked environments.

When calculating energy consumption, we first vary the bit rate as $r \in \{480, 800, 1500, 5000\}$ kb/s to calculate the number of bits transmitted within each stream. $r_0 = 800$ kb/s represents the current default rate;[6] higher rates show currently available, and potential future encoding rates. We use constant bit rate encoding, which means that the number of bits transmitted within a stream is proportional to the encoding rate.[7] To calculate the actual cost per bit transmitted, we use a variety of values to capture the many possible network setups. Specifically, we experiment with $E^b \in \{75/8, 75/4, 75/2, 75\}$ μJ, to see the effects over four (binary) orders of magnitude. We do not consider $E^b = 2$ μJ, the lowest value in the Baliga *et al.* [7], because when $E^b = 2$ μJ, $P^{\mathrm{IP}} < P^{\mathrm{DVR}}$ for the bit rates we consider, making streaming greener than recording.

[6]http://www.bbc.co.uk/blogs/bbcinternet/2009/04/bbc_iplayer_goes_hd_adds_highe.html. However, when operating in full-screen mode on modern laptops, BBC iPlayer is seen to switch to 1500 kb/s.

[7]The impact of changing to variable bit rate (VBR) encoding would also be negligible because, on average, the file size (and therefore stream size) will be a product of the video length and encoding rate (although the rate will vary over time).

The amount of content that can be offloaded depends on the storage available on individual users' DVRs. Many current DVRs may have a 500-GB or 1-TB hard disk. Standardized technical specifications such as YouView DVR specify a minimum of 320 GB [36]. However, users also need this space for manually set recordings. Therefore, we assume that SCORE has access to a small fixed-size partition in this space. As a baseline, we assume that a storage of $S_0 = 32$ GB is available, similar to the size of "reserved" partitions in architectures such as YouView [36]. We refer to this as the *constant S* case. As the content encoding bit rate increases, fewer content items can be stored in a fixed-size partition, leading to decreased gains. Therefore, we also experiment with a *rate-proportional S* case, where the partition size is taken as proportional to the bit rate encoding r as $S = S_0(r/r_0)$.

B. Understanding Energy Savings

The energy benefits are quantified by computing the metric $Energy\ Savings = ((E^{\mathrm{IP}} - E^{\mathrm{SCORE}})/E^{\mathrm{IP}}) \cdot 100$, where E^{IP} is the energy consumption of streaming all the contents and E^{SCORE} is the energy consumption using SCORE.

We wish to understand energy savings at two levels. First, we quantify the theoretical potential of content offloading. Second, we measure the savings achieved by SCORE.

1) Oracle-Based Savings: To understand the full potential of content offloading, we consider the best-case scenario for a personalized solution: An oracle that has full knowledge of future content consumption decides what to offload. Every item stored is guaranteed to be watched by the user. In this scenario, the achievable savings are limited only by the storage available.

Fig. 8 shows the results, for different combinations of parameter settings.[8] Note that the use of constant bit rate encoding means that the different encoding rates have a linear relationship. The energy savings metric depends on E^b and r, which determine the power consumed by the IP streaming option, and S, which determines the amount of content that can be offloaded. Only those combinations where inequality (2) holds are considered; combinations of low r and E^b, known to result in negative energy savings, are not shown. In general, as E^b and r increase, IP streaming consumes more energy, and the energy savings are higher. However Fig. 8(a) shows that for very high bit rates, storage can become a limiting factor: The oracle is not able to store as many items as possible at lower bit rates, resulting in smaller energy savings (e.g., at $E^b = 75$ μJ, the savings from $r = 5000$ kb/s is smaller than savings from lower bit rates). Fig. 8(b) shows that this limitation is overcome when the storage is proportional to bit rate encoding. Fig. 8(c) shows the maximum savings achievable, by removing all storage constraints (i.e., $S = \infty$). If every item can be stored locally when broadcast, up to 97% savings can be achieved at high r and E^b. The maximum savings are ≈75% considering a constant storage $S = S_0 = 32$ GB, and ≈90% considering a rate-proportional S.

2) Energy Savings in SCORE: Next, we study the savings achieved by SCORE, given access to $S_0 = 32$ GB.[9] Fig. 9 performs a sensitivity analysis and shows the average energy savings by using SCORE for different combinations of parameter

[8]Error bars in all figures show 95% confidence intervals.

[9]Due to space constraints, only the more challenging constant S case is presented for SCORE energy and traffic savings.

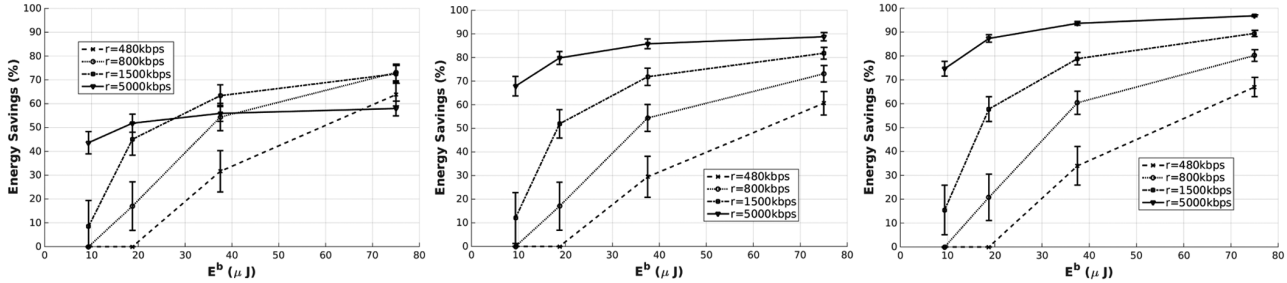


Fig. 8. Average energy savings (%) with oracle for different E^b, r, and S parameter combinations. (a) Constant S. (b) Rate-proportional S. (c) No storage constraints ($S = \infty$).

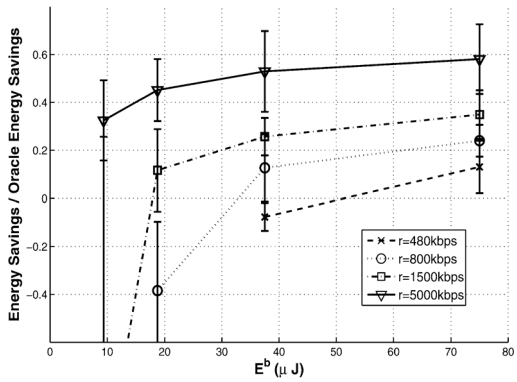


Fig. 9. Energy savings of SCORE relative to oracle. Parameter combinations where Internet streaming is more energy-efficient than DVR recording (i.e., $E^b * r <= P^{\mathrm{DVR}}$) are omitted since SCORE (similarly oracle) would not record content in settings guaranteed to waste energy.

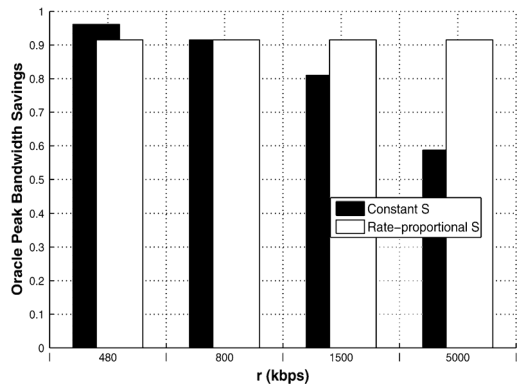


Fig. 10. Peak bandwidth savings of oracle.

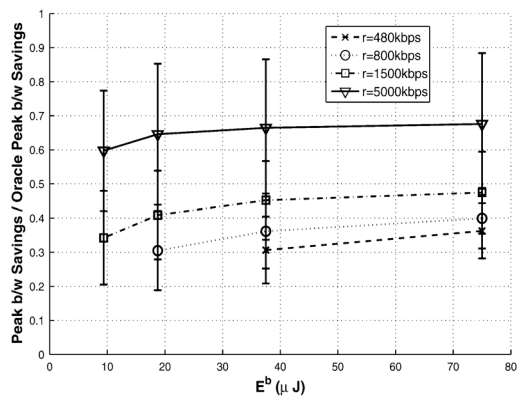


Fig. 11. SCORE peak bandwidth savings relative to oracle.

choices. For low values of r and E^b, the achievable energy savings are small, and errors in speculatively recording items not watched later can lead to negative energy savings. However, at higher bit rates, savings appear to be relatively insensitive to the assumed values of E^b and SCORE can recover 40%–60% of the optimal savings achieved by the oracle.

C. Understanding Traffic Savings

Next, we study traffic savings by computing the metric: $Peak\ bandwidth\ savings = (Q_{95}^{\mathrm{IP}} - Q_{95}^{\mathrm{SCORE}})/Q_{95}^{\mathrm{IP}}$, where Q_{95}^{SCORE} and Q_{95}^{IP} are the 95th percentile bandwidth taken across 5-min intervals by using SCORE and by streaming all the contents, respectively. This metric is intended to approximate the reductions in operating costs for ISPs, which often rely on 95th percentile bandwidth pricing. We compute the savings across the entire trace, and therefore the figure may be seen as representative of the savings for the content provider or its content delivery network (CDN) affiliate. Similar results are obtained by replacing the 95th percentile with average traffic savings, and also at the level of individual autonomous system or AS (these results omitted due to space constraints).

1) Oracle-Based Savings: Fig. 10 shows the traffic savings obtained using an oracle with complete knowledge of future requests. Unlike the energy savings computation, the oracle-based traffic savings do not depend on E^b, but only on r, the bit rate encoding, which determines the size of the IP flow, and S, the storage available on the DVR, which determines the amount of content that can be offloaded; an oracle with infinite storage can offload *all* the traffic. Thus, we only study the variation in savings for different values of r and finite values of S. The figure highlights that peak bandwidth is insensitive to the bit rate for rate-proportional S because the memory size per content item remains constant across bit rates. Fig. 10 shows that the peak bandwidth savings can be up to 96% (i.e., peak bandwidth with the oracle can be as low as 4% of the peak without oracle-based offloading), but the peak bandwidth savings rapidly decreases when storage becomes a constraint (constant S scenario, for higher bandwidths).

2) Traffic Benefits From SCORE: Fig. 11 shows a sensitivity analysis of the peak bandwidth savings obtained by SCORE for different parameter settings. Note that unlike the oracle case, the savings with SCORE depend on E^b as well as r and S. This is because the items to download are decided as a *side effect* of saving energy [(3), also see discussion in Section VI-B]. As with energy, SCORE typically recovers ≈40%–60% of the traffic savings achieved by the oracle, using 32 GB storage.[6] These savings are relatively insensitive to E^b.

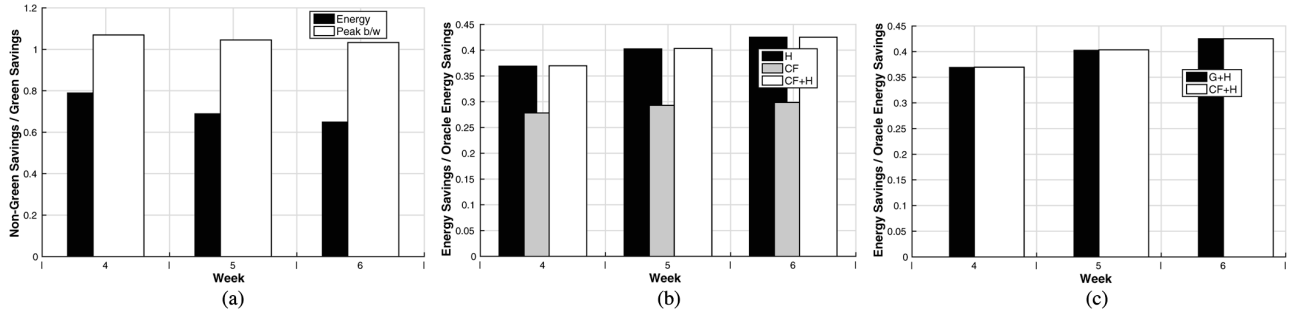


Fig. 12. Performance of "natural" alternatives in optimization and prediction. Parameters used: ($E^b = 75$ μJ, $r = 1500$ kb/s, $S = 32$ GB). (a) Optimizing energy (green) vs optimizing Traffic (non-green) savings. The green variant incurs 1.05–1.15 times more traffic than the non-green version. However, green also saves 40% more energy than non-green. (b) History versus collaborative filtering. Collaborative filtering (CF) does not offer any significant energy savings benefit over just history (H). (c) Collaborative filtering versus genres. Privacy-preserving recommender using only genre affinity ($G + H$) performs similarly to collaborative filtering ($CF + H$).

TABLE I
INDISCRIMINATELY RECORDING MOST POPULAR n ITEMS FOR EVERY USER LEADS TO NEGATIVE ENERGY SAVINGS RELATIVE TO STREAMING FROM THE INTERNET ($E^b = 75$ μJ, $r = 800$ kb/s, $S = 32$ GB, WEEK 6)

n:	1	10	20	50	100
% savings:	3.3%	4.6%	**-5.7%**	**-38.1%**	**-99.0%**

VI. "NATURAL" DESIGN ALTERNATIVES

The generic SCORE approach presented in Section IV consists of an optimizer that decides to speculatively record items based on weighting factors assigned by a predictor. However, the specific version evaluated in Section V uses a personalized optimizer for each user, which attempts to minimize the energy consumed by the user's content access needs, using knowledge of previously watched programs. Alternatives to the design presented above can be generated by using different optimization functions or predictors that yield different weighting factors. We illustrate this by considering three "natural" design variants: First, we study a nonpersonalized version, where the same weighting factor is generated for each user, based on program popularity. Next, we consider a different optimizer that aims to reduce traffic in the network, arguably a more "natural" goal. Finally, we consider how to assign weighting factors for programs not watched previously by the user. In each case, we highlight why the design we presented earlier departs from these expected "natural" choices.

A. Understanding the Need for Personalization

As a baseline, we first study a simple and straightforward approach to content offloading: offloading the most popular content to all users. Table I shows that doing so can lead to large numbers of unwatched items; recording items not watched wastes energy, resulting in decreased energy savings as n is increased. We see a net energy loss for $n = 20$ and beyond, motivating the need for a personalized, user-specific solution as developed by SCORE. Sections V-B-2 and V-C-2 show that our personalized solution can perform better than the best performing baseline: saving the most popular 10 items for every user (*top10* in Table I).

B. Traffic Optimization

As previously discussed, SCORE is optimized for energy efficiency. This can result in suboptimal traffic savings because storage capacity might not be used if the energy cost is too high. Our second design alternative therefore considers the implications of optimizing for traffic costs alone.

To achieve this, SCORE should speculatively record items regardless of energy costs. We evaluate this "price of green," by changing the optimizer to the following "non-green" version, which purely minimizes the probability that a recorded content is not watched

$$\text{minimize} \sum_{i \in C} \pi_{p_i} \cdot (1 - x_i) \tag{6}$$

subject to the memory constraint (4).

Fig. 12(a) shows the impact of greening on the energy and traffic savings in terms of the ratio of the savings achieved in the energy-aware or "green" case considered previously (3) to the savings achieved using the "non-green" case (6). The black bars show that the green solution saves up to 40% more energy compared to the non-green solution. The white bars highlight that using energy-unaware SCORE, we could only achieve a traffic savings that is about 1.05 times greater, for the parameter settings indicated. This gap would be bigger if we consider lower values of E^b. It is worth highlighting that different users can freely choose different options, optimizing for traffic or energy, since SCORE operates solely on the user's device.

C. Speculatively Recording New Program Recommendations

Up until now, we have employed a relatively simple history-based algorithm to inform SCORE. Although our evaluations show its effectiveness, the predictor of (5) cannot assign nonzero weights to new programs previously unwatched by the user. Similarly, this cannot be used for one-off programs such as movies. Next, we explore new weighting models that allow such predictions to be made.

1) Collaborative Filtering Weighting Model (CF): Our first approach is based on the same intuition as recommender systems: that new programs explored by users will be similar to programs watched in the past. Therefore, to recommend new programs to speculatively record, historical data about pairwise similarities between programs are captured as a global parameter matrix Γ. The prediction task is to use this global prior information to perform a Bayesian inference of future probabilities of watching a programs for each user. We develop a latent variable probabilistic model parameterized by Γ to perform this inference. Because it is parameterized by the program-program similarity matrix Γ, this amounts to an item-item collaborative filtering approach similar to [4], [28].

Formally, let, $U_{\mathrm{H}}, U_{\mathrm{F}}$ denote latent multinomial (categorical) random variables for a user's history and future programs, respectively. These random variables can take on 1-of-K states, each state corresponding to a different program. Let Y_{H} denote the recorded historical data (programs watched by the user). The probabilistic model is then given by

$$p(U_{\mathrm{H}}, U_{\mathrm{F}}, Y_{\mathrm{H}}|\Gamma) \propto p(Y_{\mathrm{H}}|U_{\mathrm{H}}, U_{\mathrm{F}})\Psi(U_{\mathrm{H}}, U_{\mathrm{F}}|\Gamma) \quad (7)$$

or making the assumption that the recorded history Y_{H} is dependent only on U_{H}

$$p(U_{\mathrm{H}}, U_{\mathrm{F}}, Y_{\mathrm{H}}|\Gamma) \propto p(Y_{\mathrm{H}}|U_{\mathrm{H}})\Psi(U_{\mathrm{H}}, U_{\mathrm{F}}|\Gamma). \quad (8)$$

In the above, $p(Y_{\mathrm{H}}|U_{\mathrm{H}})$ is the program likelihood, which we compute as

$$p(Y_{\mathrm{H}}|U_{\mathrm{H}}) \triangleq \begin{cases} 1, & \text{if } U_{\mathrm{H}} \subset Y_{\mathrm{H}}, \\ 0, & \text{otherwise.} \end{cases} \quad (9)$$

Similarly, $\Psi(U_{\mathrm{H}} = i, U_{\mathrm{F}} = j|\Gamma)$ is the prior belief between the history and future programs that we define as

$$\Psi(U_{\mathrm{H}} = i, U_{\mathrm{F}} = j|\Gamma) \triangleq \gamma_{i,j}, \qquad i, j \in 1:K \quad (10)$$

where, $\gamma_{i,j}$ is the i, j entry in the Γ parameter matrix. In this work, Γ is computed using historical data as $\gamma_{i,j} = |P_i \cap P_j|$, where P_i, P_j are the sets of the users watching programs i and j, respectively. Thus, Γ attempts to capture global prior information of correlations (similarities) between programs.

The final task is to infer user-specific posterior probabilities of watching different programs in the future F, given the history of recorded observations Y_{H}. Using Bayes's rule

$$p(U_F|Y_{\mathrm{H}}, \Gamma) \propto \sum_{\mathrm{H}} p(U_{\mathrm{F}}, U_{\mathrm{H}}|Y_{\mathrm{H}}, \Gamma). \quad (11)$$

By performing the summation on the right-hand side (RHS), the posterior predictive probability for a program k and user u is

$$\pi_k^{\mathrm{CF}}(u) = p(U_{\mathrm{F}} = k|Y_{\mathrm{H}}, \Gamma) = \frac{\sum_{j \in Y_{\mathrm{H}}} \gamma_{k,j}}{Z} \quad (12)$$

where $Z = \sum_{k \in 1:K} \sum_{j \in Y_{\mathrm{H}}} \gamma_{k,j}$ is a normalization factor.

It is natural to combine the benefits of our initial model, (5), which accurately assigns high weights for episodes of programs regularly watched by a user, with the second model (12), which can assign nonzero weights to new programs. Thus, we get a new weighting factor $CF + H$

$$\pi_p^{\mathrm{CF+H}}(u) = \max\left(\pi_p^{\mathrm{H}}(u), \pi_p^{\mathrm{CF}}(u)\right). \quad (13)$$

2) Privacy Preserving Recommendations $(G + H)$: CF and CF+H require a central server to collect and retain information about all users' viewing patterns to create the global matrix Γ. Although this is done inherently in iPlayer's current streaming model, it will not be the case with SCORE, which records autonomously from the broadcast interface. Consequently, we must sacrifice some degree of privacy to implement a CF strategy. We therefore extend this to offer a local content-based filtering approach that does not require a user to reveal viewing history.

Our content-based filtering model weights each program based on the affinity of the user to the genre(s) of the program. We adopt a vector space approach and assign to each user u a vector $\mathbf{g}_u = (g_u^1, g_u^2, \ldots, g_u^m)$, where g_u^j is the number of content items of the jth genre watched by the user. Similarly, each program p is assigned a vector $\mathbf{g}_p = (g_p^1, g_p^2, \ldots, g_p^m)$, where g_p^j is the number of episodes of p tagged with the jth genre. The genre-based weight π_p^{G} is then calculated as the cosine similarity between the user's genres and the genres of the program

$$\pi_p^{\mathrm{G}}(u) = \frac{\mathbf{g}_u \cdot \mathbf{g}_p}{\|\mathbf{g}_u\| \, \|\mathbf{g}_p\|}. \quad (14)$$

As before [e.g., (13)], we combine this with the user's personal history (which can be computed and kept locally on the user's DVR, and thus does not compromise privacy)

$$\pi_p^{\mathrm{G+H}}(u) = \max\left(\pi_p^{\mathrm{H}}(u), \pi_p^{\mathrm{G}}(u)\right). \quad (15)$$

3) Evaluating Program Recommendation Extensions: We evaluate these new weighting models by randomly selecting 27 459 users from our traces, who watched at least 2 programs a week (to allow program-program similarity to be calculated). Fig. 12(b) compares this against our original history-based weighting model H. It presents the energy savings, and the overall traffic savings, as defined by $(T^{\mathrm{SCORE}} - T^{\mathrm{IP}})/T^{\mathrm{IP}}$, where T^{SCORE} and T^{IP} are the amount of streamed traffic by using SCORE and by streaming all the watched content, respectively.

It can be seen that CF by itself performs poorly, suggesting that users' content consumption patterns are dictated more by history (i.e., watching different episodes of the same programs), rather than by exploring new programs. Indeed, even $CF + H$ does not offer any significant benefits over the much simpler weighting factor H. Fig. 12(c) shows that the privacy-preserving model $G + H$ performs similarly to $CF + H$, suggesting that simple models may be sufficient to incorporate recommendations for speculatively recording new programs not watched before. Of course, results for H are limited to corpora that are serial-based. The BBC, and most terrestial TV channels in the UK, have a heavy bias towards serial content, which is why H is so effective. Although these channels do serve non-serial content, this does not achieve the popularity of their serialized counterparts. This means that SCORE would be effective at serving most TV channels, excluding those specializing in one-off shows, e.g., movies. Our future work will involve looking at the performance of these weighting models for different corpora.

VII. Related Work

A number of seminal works [3], [11], [16], [20], [37] have examined different forms of (video) delivery over the Internet. These range from walled garden IPTV architectures to P2P live streaming workloads. We add to this list by examining a catch-up TV workload. Here, we focus on push- versus pull-style accesses. Previously, we have also examined the factors affecting adoption and usage of TV streaming across the UK ISP ecosystem [25]. In comparison to the previous largest measurement study of catch-up TV [3], our work makes new observations on push versus pull access patterns, includes radio workloads in addition to TV, and proposes SCORE as a novel mechanism to mitigate the footprint of catch-up. Our dataset also contains orders of magnitude more users.

The key contribution of our work has been a novel approach to combining the benefits of push and pull content delivery. This has been driven by an optimizer targeted at reducing energy costs. It has been recognized before that a large amount of savings can be realized by offloading content from the servers [21]. In walled-garden IPTV approaches, when the operator has control over the network, caching at appropriate

locations and branch points within the network can be effective [6], [9], [34]. Deployments operating over the public Internet have to rely on end-users, and a popular strategy is to use P2P approaches where users collaboratively download from each other to decrease server load. However, supporting the delivery constraints of streaming in P2P architectures typically introduces complexity such as elaborate mesh/tree topology construction (e.g., [10] and [26]), or careful chunk-scheduling strategies (e.g., [5], [13], [22], and [35]). Instead of peers, SCORE exploits the existing *broadcast channel* to decrease server and network load. While this makes the SCORE solution specific to catch-up TV/radio, it also makes the design straightforward. Recently, we have shown that peer-assisted CDNs can also be effective for catch-up TV [24].

Prefetching content is a common trick in CDNs (e.g., [9], [23], [33], and references therein). However, most such works that consider delivering large objects such as videos need to balance the bandwidth consumed by speculative prefetching with the potential benefits. Instead, SCORE uses a cheaper, out-of-band distribution channel (DTT), and hence can replicate freely, subject only to storage constraints. In this respect, SCORE is similar to offloading from 3G/4G onto cheaper Wi-Fi networks (e.g., [19] and [27]). However, mobile data offloading schemes typically involve delaying access until Wi-Fi becomes available, whereas with SCORE, content is prefetched and therefore immediately available. Importantly, Wi-Fi allows fetching data using user-specific request/response streams, whereas SCORE operates over a broadcast delivery mechanism common to all users. This allows the benefits of SCORE to accrue not only to users and access networks, but also the core and also decreases the content provider's network costs. Recent work explores the use of cellular broadcast channels (e.g., in LTE) to broadcast popular objects [18]. However, recording the top-n items could lead to negative energy savings (c.f., Table I). SCORE exploits semantic knowledge of access patterns to *catch-up* videos (e.g., serial affinity), to make more informed, personalized decisions. Our focus on decreasing *system-wide* energy footprint (rather than just on mobile phones) is also a distinguishing factor.

Functionality similar to SCORE is available on some commercially available DVRs, but there are differences. For example, some DVRs, such as TiVo, assist in content discovery by recommending *new* programs to watch [32]. Our goal is similar, but with an important difference: We wish to learn the *existing* viewing habits of users and anticipate their usage of catch-up TV. TiVo essentially records as many relevant suggestions as possible, as low-priority items to be erased if user-requested recordings require space. SCORE is much more conservative because recording content not watched later on wastes energy. Recent commercial offerings in the US such as "Primetime Anytime" (c.f., http://dishuser.org/ptat.php) from DISH, automatically record evening prime time shows for the four major broadcast networks during evening Prime Time. Sky TV in the UK follows a similar approach. The programs recorded by these offerings are expected to be the most popular shows. However, as discussed above, this could lead to negative energy savings.

VIII. Discussion and Conclusion

We are currently witnessing the long-predicted convergence of IP and media networks in various forms. While this has offered additional functionality such as catch-up TV, the encroaching of broadcast media on the IP network can lead to additional network traffic and energy consumption.

Our contributions are twofold. First, we have explored the key differences between traditional broadcast (push) and emerging pull-based models of delivery. These observations led us to our second contribution: a simple approach that can leverage both broadcast push and online pull—the Speculative Content Offloading and Recording Engine (SCORE). SCORE exploits the predictable nature of users' content consumption patterns to reduce the energy and network footprint of catch-up TV. Our evaluation using traces from BBC iPlayer showed that significant energy savings can be achieved (up to 77%) while also reducing the network footprint. We believe that the results are robust, given the scale of our trace. The results may be also generalizable to other catch-up TV systems (e.g., iView in Australia, Hulu in the US, or 4oD and ITV Player in the UK), which all share similar access patterns such as a dominance of serialized TV shows.

Our main motivation in developing SCORE was to demonstrate that it is relatively easy to offload catch-up video streams from the Internet. Various future avenues of work exist for expanding upon this concept. There is great potential for developing more sophisticated prediction algorithms. Although we experimented with this, we did not find notable savings over SCORE's simple history-based approach. Future work would therefore need to focus on exploiting alternative information sources, e.g., content ratings or social network information. A second avenue of future work would be to develop optimization algorithms that focus on different considerations, e.g., content provider preferences or ISP costs.

References

[1] "Commission Regulation No 107/2009 of 4 February 2009 implementing directive 2005/32/EC of the European Parliament and of the Council with regard to ecodesign requirements for simple set-top boxes," *Official J. Eur. Union L36*, pp. 8–14, 2009.

[2] NorDig, "NorDig unified requirements for integrated receiver decoders for use in cable, satellite, terrestrial and IP-based networks," Ver. 2.2.1, 2010.

[3] H. Abrahamsson and M. Nordmark, "Program popularity and viewer behaviour in a large TV-on-demand system," in *Proc. IMC*, 2012, pp. 199–210.

[4] K. Ali and W. Van Stam, "TiVo: Making show recommendations using a distributed collaborative filtering architecture," in *Proc. ACM KDD*, 2004, pp. 394–401.

[5] S. Annapureddy *et al.*, "Is high-quality VoD feasible using P2P swarming?," in *Proc. WWW*, 2007, pp. 903–912.

[6] D. Applegate *et al.*, "Optimal content placement for a large-scale VoD system," in *Proc ACM CoNEXT*, 2010, Art. no. 4.

[7] J. Baliga, R. Ayre, K. Hinton, W. V. Sorin, and R. S. Tucker, "Energy consumption in optical IP networks," *J. Lightw. Technol.*, vol. 27, no. 13, pp. 2391–2403, Jul. 2009.

[8] BBC, "An introduction to corporate responsibility at the BBC," 2010 [Online]. Available: http://downloads.bbc.co.uk/outreach/bbc-corporate-responsibility.pdf

[9] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *Proc. IEEE INFOCOM*, 2010, pp. 1–9.

[10] M. Castro *et al.*, "SplitStream: High-bandwidth multicast in cooperative environments," in *Proc. SOSP*, 2003, pp. 298–313.

[11] M. Cha, P. Rodriguez, J. Crowcroft, S. Moon, and X. Amatriain, "Watching television over an IP network," in *Proc. IMC*, 2008, pp. 71–84.

[12] J. Chandaria, J. Hunter, and A. Williams, "A comparison of the carbon footprint of digital terrestrial television with video-on-demand," BBC Research Whitepaper 189, Mar. 2011.

[13] Y. Choe, D. Schuff, J. Dyaberi, and V. Pai, "Improving VoD server efficiency with bittorrent," in *Proc. ACM MM*, 2007, pp. 117–126.

[14] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. Hoboken, NJ, USA: Wiley, 2006.
[15] Deloitte, "Technology, media & telecom predictions," May 2012.
[16] F. Dobrian *et al.*, "Understanding the impact of video quality on user engagement," in *Proc. ACM SIGCOMM*, 2011, pp. 362–373.
[17] S. Eastman and D. Ferguson, *Media Programming: Strategies and Practices*. Boston, MA, USA: Cengage Learning, 2012.
[18] A. Finamore *et al.*, "Is there a case for mobile phone content pre-staging?," in *Proc ACM CoNEXT*, 2013, pp. 321–326.
[19] B. Han *et al.*, "Mobile data offloading through opportunistic communications and social participation," *IEEE Trans. Mobile Comput.*, vol. 11, no. 5, pp. 821–834, May 2012.
[20] X. Hei, C. Liang, J. Liang, Y. Liu, and K. Ross, "A measurement study of a large-scale P2P IPTV system," *IEEE Trans. Multimedia*, vol. 9, no. 8, pp. 1672–1687, Aug. 2007.
[21] C. Huang, J. Li, and K. W. Ross, "Can Internet video-on-demand be profitable?," in *Proc. ACM SIGCOMM*, 2007, pp. 133–144.
[22] Y. Huang *et al.*, "Challenges, design and analysis of a large-scale P2P-VoD system," in *Proc. ACM SIGCOMM*, 2008, pp. 375–388.
[23] J. Kangasharju, J. Roberts, and K. W. Ross, "Object replication strategies in content distribution networks," *Comput. Commun.*, vol. 25, no. 4, pp. 376–383, 2002.
[24] D. Karamshuk, N. Sastry, J. Chandaria, and A. Secker, "ISP-friendly peer-assisted on-demand streaming of long duration content in BBC iPlayer," in *Proc. IEEE INFOCOM*, 2015.
[25] D. Karamshuk, N. Sastry, J. Chandaria, and A. Secker, "On factors affecting the usage and adoption of a nation-wide TV streaming service," in *Proc. IEEE INFOCOM*, 2015.
[26] D. Kostić, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: High bandwidth data dissemination using an overlay mesh," in *Proc. SOSP*, 2003, pp. 282–297.
[27] K. Lee, J. Lee, Y. Yi, I. Rhee, and S. Chong, "Mobile data offloading: How much can WiFi deliver?," in *Proc. ACM CoNEXT*, 2010, pp. 425–426.
[28] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Comput.*, vol. 7, no. 1, pp. 76–80, Jan.–Feb. 2003.
[29] Neilsen, "Report: Bigger TVs, DVR and Wi-Fi among hot U.S. home technology trends," 2010.
[30] Ofcom, "Communications market report 2012," Jul. 2012 [Online]. Available: http://stakeholders.ofcom.org.uk/binaries/research/cmr/cmr12/CMR_UK_2012.pdf
[31] Sandvine, "Global Internet Phenomena Report, 1 h 2012," May 2012.
[32] TiVo, "How to find great new shows with TiVo suggestions," Accessed Apr. 25, 2012 [Online]. Available: http://www.tivo.com/mytivo/howto/getthemostoutoftv/howto_use_suggestions.html
[33] A. Venkataramani *et al.*, "The potential costs and benefits of long-term prefetching for content distribution," *Comput. Commun.*, vol. 25, no. 4, pp. 367–375, 2002.
[34] M. Verhoeyen, D. De Vleeschauwer, and D. Robinson, "Content storage architectures for boosted IPTV service," *Bell Labs Tech. J.*, vol. 13, no. 3, pp. 29–43, 2008.
[35] X. Yang, M. Gjoka, P. Chhabra, A. Markopoulou, and P. Rodriguez, "Kangaroo: Video seeking in P2P systems," in *Proc. Int. Conf. Peer-to-Peer Syst.*, 2009, p. 6.
[36] YouView TV, Ltd., "YouView core technical specification," 2011.
[37] H. Yu, D. Zheng, B. Zhao, and W. Zheng, "Understanding user behavior in large-scale video-on-demand systems," *Oper. Syst. Rev.*, vol. 40, no. 4, pp. 333–344, 2006.

Gianfranco Nencioni received the Master's degree in telecommunication engineering (*cum laude*) and Ph.D. degree in information engineering from the University of Pisa, Pisa, Italy, in 2008 and 2012, respectively.

In the Fall of 2011, he was a visiting Ph.D student with the Computer Laboratory, University of Cambridge, Cambridge, U.K., supervised by Prof. Jon Crowcroft. From 2012 to 2015, he was a Postdoctoral Fellow with the Department of Information Engineering, University of Pisa. He is currently a Postdoctoral Fellow with the Department of Telematics, Norwegian University of Science and Technology (NTNU), Trondheim, Norway. His past research activity has mainly regarded energy-aware routing and network design in both wired and wireless environments. His current research activity regards dependability on SDN and NFV.

Nishanth Sastry (S'09–M'11) received the Bachelor's degree from Bangalore University, Bangalore, India, in 1999, the Master's degree from The University of Texas at Austin, Austin, TX, USA, in 2001, and the Ph.D. degree from the University of Cambridge, Cambridge, U.K., in 2011, all in computer science.

He is a Senior Lecturer with King's College London, London, U.K. He has over 6 years of experience in the industry (Cisco Systems, Bangalore, India, and IBM Software Group, Westford, MA, USA) and industrial research labs (IBM T. J. Watson Research Center, Cambridge, MA, USA). His work has ranged over several layers of the network stack, and he is currently involved in building better networked systems by harnessing social network information.

Dr. Sastry's honors include a Best Undergraduate Project Award, a Best Paper Award from the Computer Society of India, a Yunus Innovation Challenge Award at the Massachusetts Institute of Technology IDEAS Competition, a Benefactor's Scholarship from St. John's College, Cambridge, U.K., a Cambridge Philosophical Society Research Studentship, a Cisco Achievement Program Award, and several awards for his work at IBM.

Gareth Tyson received the Ph.D. degree in networking and distributed systems from Lancaster University, Lancaster, U.K., in 2010.

He is a Lecturer with Queen Mary, University of London, London, U.K. His research interests include Internet measurements, content distribution, and the future Internet.

Vijay Badrinarayanan received the bachelor's degree in electronics and communication engineering from Bangalore University, Bangalore, in 2001, the M.S. degree in electrical engineering from the Georgia Institute of Technology, Atlanta, GA, USA, in 2006, and the Ph.D. degree in signal and image processing from INRIA, Rennes, France, in 2009.

He is currently a Senior Research Associate with the Machine Intelligence Laboratory, Department of Engineering, University of Cambridge, Cambridge, U.K. His research interests are in probabilistic graphical models, deep machine learning applied to computer vision, data modeling, and analysis problems.

Dmytro Karamshuk (M'15) is a Postdoctoral Research Associate with King's College London, London, U.K. His research is focused on studying behavior of Internet users to assist design of future generation communication networks.

Jigna Chandaria received the M.Eng. degree in electrical and information sciences from the University of Cambridge, Cambridge, U.K., and the M.Sc. degree in innovation and design for sustainability from Cranfield University, Cranfield, U.K.

She is a Lead Research Engineer with BBC R&D, London, U.K. Her research focuses on the environmental impact of broadcasting and media technology. She is interested in how we assess environmental impact, sustainable design, and using sustainability to drive innovation. She chairs the EBU Strategic Programme on Sustainable Technology in Broadcasting. Prior to starting up this research area, she developed new technology for TV production, in particular real-time 3-D graphics for sports and virtual studios. Her areas of research included real-time image processing, camera tracking, augmented reality, and 3-D modeling. She worked closely with BBC production teams, worked on several U.K. and European collaborative projects, and contributed to the Piero sports graphic system, which was awarded The Queen's Award for Enterprise.

Jon Crowcroft (SM'95–F'04) received the B.S. degree in physics from Trinity College, University of Cambridge, Cambridge, U.K., in 1979, and the M.Sc. degree in computing and Ph.D. degree from University College London (UCL), London, U.K., in 1981 and 1993, respectively.

He has been the Marconi Professor of Communications Systems with the Computer Laboratory, University of Cambridge, since 2001. He has worked in the area of Internet support for multimedia communications for over 30 years. He likes teaching and has published a few books based on learning materials. Three main topics of interest have been scalable multicast routing, practical approaches to traffic management, and the design of deployable end-to-end protocols. Current active research areas are opportunistic communications, social networks, and techniques and algorithms to scale infrastructure-free mobile systems. He leans towards a "build and learn" paradigm for research.

Prof. Crowcroft is a Fellow of the Royal Society, the Association for Computing Machinery (ACM), the British Computer Society, the IET, and the Royal Academy of Engineering.