Enhancing Video Accessibility and Availability Using Information-Bound References

Ashok Anand, Athula Balachandran, Aditya Akella, Vyas Sekar, and Srinivasan Seshan

***Abstract*—Users are often frustrated when they cannot view video links shared via blogs, social networks, and shared bookmark sites on their devices or suffer performance and usability problems when doing so. While other versions of the same content better suited to their device and network constraints may be available on other third-party hosting sites, these remain unusable because users cannot efficiently discover these and verify that these variants match the content publisher's original intent. Our vision is to enable consumers to leverage *verifiable alternatives* from different hosting sites that are best suited to their constraints to deliver a high quality of experience and enable content publishers to reach a wide audience with diverse operating conditions with minimal upfront costs. To this end, we make a case for *information-bound references* or IBRs that bind references to video content to the underlying information that a publisher wants to convey, decoupled from details such as protocols, hosts, file names, or the underlying bits. This paper addresses key challenges in the design and implementation of IBR generation and resolution mechanisms, and presents an evaluation of the benefits IBRs offer.**

I. Introduction

A SIGNIFICANT and growing fraction of Internet traffic today consists of video content [5]. Many users discover and access such content via links shared through traditional (e.g., email, IM) and social media applications (e.g., online social networks, blogging services, and social bookmarking sites [36], [10]. Unfortunately, the URLs used to share videos are fragile as they are inherently bound to a specific protocol, host, and file name [32]. This tight coupling is especially problematic as users want to access content from an increasingly heterogeneous set of the device (e.g., smartphones, tablets) and network conditions. As shown in Section II, this results in significant accessibility (e.g., content not playable) and quality-of-experience (QoE) problems (e.g., high buffering or start-up latencies for video [31]).

Fortunately, measurement studies show that there are alternative versions of the same content (e.g., resolutions, formats) available on different third-party hosting sites [4], [35]. We could potentially alleviate the above accessibility and QoE issues if we had a mechanism that enables users to find the alternative best suited to their current software, device, and network context. A variety of entities, including device vendors, search engine providers, social networking and bookmarking sites, would have strong incentives to deploy such a mechanism (Section II).

However, it is challenging to realize such a mechanism in practice. To see why, consider two seemingly natural strawman approaches. To delay the binding to a specific video file, one option for content publishers is to only provide keywords. Content consumers can then use search engines to find suitable alternatives. Unfortunately, search keywords are notoriously *contention prone* and give users no confidence that the result matches the publisher's intent [32]. Alternatively, one could envision new *data-centric* architectures as they decouple the data from the delivery mechanisms [29], [42]. The hashing schemes in these proposals, however, operate at the byte-level. Thus, the names for different encodings of the same content will be different and preclude opportunities for leveraging the alternative versions.

The key here is choosing an appropriate granularity at which we need to bind the intent of the publisher. Data-centric names offer one extreme at the byte-level representation. Human-readable keywords offer another point with loose bindings that are susceptible to abuse. What we ideally need is a mechanism that delays the binding sufficiently to provide the flexibility to choose alternatives, but at the same time allows clients to be assured that the variant they choose matches the publishers' original intent. We call this new type of link an *Information-Bound Reference* or *IBR* since it references the information that a user wants rather than its location, format, or file name. Using IBRs will allow content publishers to reach a broader audience without significant quality problems and without high upfront infrastructure costs. IBRs will improve consumers' quality of experience and minimize frustrations due to poor performance or content inaccessibility.

In this paper, we address key algorithmic and system design challenges in realizing an IBR-based content retrieval framework. Our key contributions are:

- **IBR Generation:** IBRs must be encoding-invariant, resilient to contention (i.e., two unique pieces of information map to different IBRs) and compact (i.e., be small relative to the content they refer to). To this end, we envision a novel use-case for multimedia fingerprinting algorithms [21], [38], [39]. Our vision, however, raises new performance, scalability, and verifiability requirements. Thus our

Manuscript received May 23, 2014; revised January 02, 2015; accepted February 12, 2015; approved by IEEE/ACM Transactions on Networking Editor A. X. Liu. Date of publication June 01, 2015; date of current version April 14, 2016. This work was supported in part by the National Science Foundation (NSF) CAREER Award (CNS-0746531), NSF NetSE grant (CNS-0905134), and a gift from Cisco.

A. Anand is with Instart Logic, Bangalore, India.

A. Balachandran, V. Sekar, and S. Seshan are with Carnegie Mellon University, Pittsburgh, PA 15213 USA.

A. Akella is with University of Wisconsin, Madison, WI 53705 USA.

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.2015.2413352

specific contribution lies in practically synthesizing these techniques to serve as IBRs (Section IV).

- **Resolution:** Users should be able to use IBRs to quickly find the copy of content that is most appropriate for their device and network conditions. What makes this challenging is that video encodings are intrinsically lossy which means that the IBR matching involves a "fuzzy" match. To this end, we design a scalable lookup service that can provide ≈1 million lookups/s on a 25-node cluster leveraging algorithms for locality-sensitive hashing [27] (Section V).
- **End-to-end realization:** We demonstrate an end-to-end realization of an IBR architecture: client-side extensions for desktop and mobile platforms, support for legacy consumers and publishers, and additional mechanisms to enhance the intrinsic verifiability offered by IBRs (Sections VI, VII).

Using a combination of public video and image datasets and end-to-end experiments, we show that IBRs are practical and offer significant benefits (Section VIII). Specifically, we show that IBRs can offer significant improvements in quality of experience: reducing the video startup delay (i.e., the time it takes for the video to start playing after the user clicks the play button) by 4–9 seconds, increasing content accessibility, and avoiding rebuffering events (from 50% of time spent buffering to zero). We also show that using IBRs are practical—web pages rewritten to use IBRs incur low load time overhead (≈0.5 s), IBRs incur close to zero false positives for content naturally occurring in the wild, and IBRs can be resolved efficiently using our system (≈1 million lookups/sec on a 25-node cluster).

II. Motivation

We show empirical evidence of quality of experience issues that users face today in accessing shared video links. We also highlight the (unrealized) promise of leveraging alternate versions of the same content from different sites.

Quality issues in shared video links: Recent analysis shows that social media applications are the dominant mode through which users discover mobile videos [10]. We collected the top-500 most popular video links posted on reddit.com and found that ≈10% of the videos are hosted by small providers (i.e., not on YouTube, Vimeo, DailyMotion).[1] Such smaller hosting sites typically do not offer multiple formats or bitrates. Table I summarizes anecdotal evidence of user experience issues on a laptop, iPhone, and an Android device for these video links. We see a diverse range of problems with excessive buffering, the video not playing, and requiring the user to manually navigate to mobile versions. In some cases, we even see problems accessing videos on the laptop, possibly because of server-side issues. Under low bandwidth connections (512 Kbps or less), we see multiple buffering events while viewing these video links as the video content providers typically do not offer lower bitrates for such conditions.

Even popular providers have QoE issues: The quality

[1]In an earlier measurement, we found ≈30% of video links were to smaller providers. Moreover, our analysis focuses on the popular content; we suspect that links to smaller providers may be even more prevalent in the "long-tail" of less popular links.

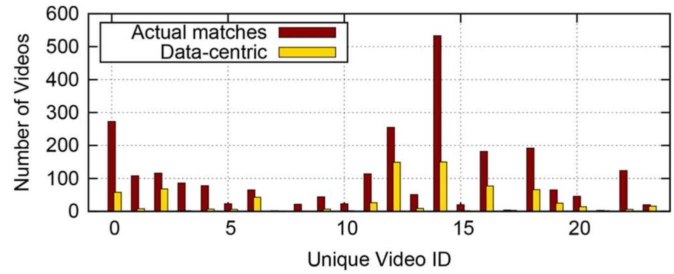

Fig. 1. Using a public video dataset, we (manually) verified that there are indeed several alternatives for the same video. We also see that data-centric approaches miss many opportunities for leveraging these alternatives as they are tightly bound to the byte-level representations.

problems are not just restricted to the lower end of the video ecosystem—even well-provisioned sites such as YouTube, DailyMotion, and Vimeo suffer user QoE and accessibility issues. We found three separate videos hosted on all providers and attempted to access these from four different devices. We focus on the startup delay (i.e., difference between the time when user hits the play button to the time the video starts to play) because it is an important measure of QoE that impacts user retention rates [24], [30]. Table II shows the startup delay across the combinations of device, video, and hosting site. First, in many cases the video was not available for mobile devices or failed to play. Second, no specific video hosting site is the best choice across all platforms; e.g., YouTube is good for laptops but not for mobile devices. Third, there is also some diversity across videos w.r.t accessibility and performance; e.g., Video1 seems to be generally accessible on the iPhone but Video2/Video3 seem to have some issues (no mobile version available or fail to play). We have also confirmed that these results can be reproduced across multiple network locations and videos (not shown for brevity).

Alternatives exist, but cannot be exploited: While the above results show evidence of QoE problems, they also suggest some hope. For instance, even though Video2 and Video3 fail with DailyMotion on the iPhone, we can view the content from Vimeo. Measurements have shown that this is indeed the case—there are alternate versions of the same content (e.g., different resolutions, formats) available on third-party sites [4], [35]. Using a public video dataset [4], we analyze the number of variants of the same video (denoted by a unique video ID) in Fig. 1 depicted as "Actual Match". The challenge, however, is that users do not have a way to automatically discover these variants. Even if they do, they may have to manually check that these alternatives meet the intent (e.g., it is not "rickrolling" them) and that these are customized to meet their device and network constraints.

There is a natural parallel here to the motivation for *data-centric* architectures—users ultimately care about the data and not who is serving it [40], [29], [42]. Thus, data-centric schemes appear to be a seemingly natural strawman solution. Using the same video dataset [4], Fig. 1 also shows the inability of data-centric approaches to identify variants of the same video. The reason is that the hashing algorithms underlying data-centric schemes operate at the byte-level. Because minor encoding differences can lead to drastically different byte-level representations, data-centric approaches cannot identify these variants as

TABLE I

ANECDOTAL EVIDENCE OF USER EXPERIENCE ISSUES ON LINKS SHARED THROUGH SOCIAL MEDIA SITES THROUGH NON-POPULAR VIDEO HOSTING SITES. OUR GOAL IS NOT TO PINPOINT SPECIFIC PROVIDERS BUT SHOW PROBLEMS SYMPTOMATIC OF THE ENTIRE VIDEO+MOBILE ECOSYSTEM. WE USE A CONTROLLED WIFI SETTING WITH THE SAME CLIENT-SIDE ACCESS BANDWIDTH. THE REPORTED PROBLEMS ARE FOUND DURING ALMOST EVERY BROWSING (TOTAL OF 10 RUNS) OF THE GIVEN VIDEO LINK ON THE GIVEN DEVICE

Provider	iPhone	Android	Laptop
comedycentral.com	Doesn't play	Has play button, but no video	OK
complex.com	Buffering	Doesn't play	Buffering
vine.co	OK	Excessive buffering	OK
ctvnews.ca	Additional click to mobile site		OK

TABLE II

VIDEO STARTUP LATENCY ACROSS POPULAR VIDEO HOSTING PROVIDERS; YT, V, AND DM STAND FOR YOUTUBE, VIMEO, AND DAILYMOTION RESPECTIVELY. THE "N/A" ENTRIES SHOW CASES WHERE THERE WAS NO MOBILE VERSION AVAILABLE. THE "FAIL" ENTRIES REPRESENT CASES WHERE THERE APPEARED TO BE A VIDEO BUT AN ERROR MESSAGE APPEARS AFTER 30 S. WE USE A CONTROLLED WIFI SETTING WITH THE SAME CLIENT-SIDE ACCESS BANDWIDTH. THE REPORTED NUMBERS ARE AVERAGED OVER 10 RUNS

Device	**Startup latency (seconds)**								
	Video1			Video2			Video3		
	YT	V	DM	YT	V	DM	YT	V	DM
iPad2	4	6	10	2	6	15	n/a	4	4
Laptop	1	1	2	1	2	4	2	2	2
iPhone4	1	3	5	1	2	fail	n/a	5	fail
Droid	5	fail	4	3	n/a	2	n/a	4	3

they only capture *exact byte-level* copies. This is also true for more flexible chunking schemes based on Rabin fingerprinting [35].

Summary: In summary, we observe that users face significant accessibility and quality of experience issues in accessing video content as the specific links are not suitable for their devices. While there are alternative versions of the desired content on other hosting sites, users cannot reliably discover and leverage these; even future data-centric architectures fail to adequately capture the available alternatives.

What we ideally need is a service that allows users to flexibly leverage third-party alternatives to minimize QoE issues. In fact, many vendors in the mobile and social media ecosystem have immediate incentives and are also naturally positioned to offer such a service. For example, device vendors or service providers can simplify and spur adoption without relying on support from content providers. Similarly, social media sites like Reddit or Facebook would also be naturally incentivized to enhance the user experience and thus maximize user retention.[2] Alternatively, search engine operators like Google or Bing may also offer such enhanced services specifically targeted to attract mobile traffic.[3]

[2]e.g., Reddit already runs some (beta) image deduplication services such as karmadecay.com.

[3]Google offers the Mobilizer service to create mobile-friendly websites.

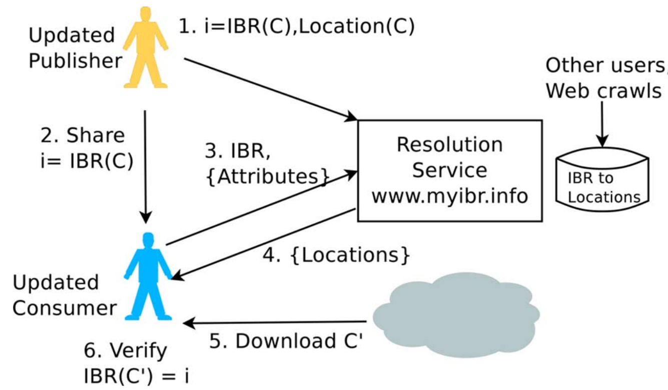

Fig. 2. A high-level overview of how IBRs can be used for sharing and retrieving multimedia content.

III. ARCHITECTURE OVERVIEW

Our overarching vision is to enable users to flexibly discover and leverage the alternative versions of the content that can deliver the best delivery performance (e.g., accessibility, bitrate, buffering, startup delay). In this section, we begin with a high-level overview of our approach and highlight the main challenges involved in realizing this vision.

Fig. 2 shows an overview of our framework. Today, publishers share multimedia content via URLs that link to hosting sites. As we discussed earlier, this constrains the accessibility and usability of this content by restricting users (e.g., their friends on social networks) to the particular hosting site. Rather than tightly couple references to sites, formats, and encodings, we envision publishers who instead post an *information-bound reference* (IBR) satisfying two key properties:

1) **encoding invariant:** different formats and resolutions of the same content map to the same IBR; and
2) **bound to the information:** unique contents should have different IBRs.

This IBR will be posted via a *resolution service (RS)*; e.g., available at www.myibr.info in Fig. 2. The RS maintains a mapping between an IBR and the locations (hosting sites) of different versions of content with this IBR. These mappings can be populated using a combination of two techniques: (1) Each publisher generates an IBR for the content she posts or references, and registers an (IBR, location) tuple with the RS; and (2) The RS crawls the Web to identify variants of the same content. For each such alternative content, the RS also registers attributes such as the format and bitrate. Note that our framework is general enough to support multiple such resolution services. As we discussed earlier, several players in the mobile/social media ecosystem (e.g., Facebook, reddit, Google, Apple) have natural incentives to deploy such a service and also motivate their users to post IBRs (rather than direct URLs) to enhance user retention and attract more "eyeballs".

Consumers query the RS (e.g., http://www.myibr.info?ibr="xyz") to find potential alternatives of the multimedia content that have the same IBR = xyz and can choose a version that best matches their constraints; i.e., a QoE aware selection. They fetch this content from a suitable hosting site, and verify that this matches the IBR = xyz. Our goal is to provide consumers a high degree of flexibility to view an alternative encoding that is well-suited to their device and current operating constraints. In the simple case, this flexibility is *static*; e.g., based on what type of device a consumer is using, what the screen size is, how long would this content take to load, and so on. More generally, this flexibility needs to be *dynamic*; e.g., switching bitrates or streaming servers due to bandwidth changes, or reducing the bitrate when the battery life becomes low.

While this vision builds on the idea of indirection to delay the binding between the content reference and the actual content served, there are two specific challenges that we need to address in order to realize this vision:

1) Building on now familiar arguments for contention-freeness and verifiability, IBRs must be algorithmically generated from the underlying content, as opposed to, say, human-input labels [32]. To this end, we identify and synthesize algorithms from multimedia fingerprinting (e.g., [21], [38]) in Section IV. We also describe how we can augment the verification guarantees in Section VI.
2) We need a scalable resolution infrastructure and efficient mechanisms for enabling clients to exploit the flexibility that IBRs offer. Specifically, the multimedia techniques to identify similar content inherently require "fuzzy" matches (i.e., they are not exact string matches) and, thus, we cannot leverage traditional techniques for building scalable key-value stores (e.g., [22]). Thus, we design a scalable resolution service building on locality sensitive hashing in Section V and describe practical client-side capabilities for QoE-aware delivery in Section VII.

IV. Generating and Sharing IBRs

In this section, we focus on the first high-level challenge: generating and posting references that are algorithmically bound to the underlying information and invariant across encodings, bitrates etc. We focus primarily on video content as it represents a dominant fraction of Internet traffic [5].

Our key insight here is that we can leverage a rich literature of techniques for multimedia fingerprinting to design IBRs [21], [20]. Multimedia fingerprinting is used in a variety of applications today: duplicate detection [38] and detecting copyright violations [14]. At a high level, this notion of identifying other multimedia content that is "close" to a given object suggests that such fingerprinting algorithms can serve as a useful starting point. However, our IBR vision raises new system-level challenges related to scalability, performance, and verifiability that do not arise in the traditional multimedia applications. In this section, we show how we synthesize these approaches to address these challenges. We begin by describing how to derive IBRs for images which form the basis for our video IBRs.

A. Building Block: Image IBRs

There are three broad classes of techniques used in image fingerprinting that are based on: (1) understanding spatial structure (e.g., [34]), (2) capturing color distributions (e.g., [33]), and (3) frequency domain analysis (e.g., [21], [9]). The first class of techniques identifies spatial gradients to mimic how the human eye recognizes objects. Color histogram techniques look at the distribution of R, G, B values in an image. These two techniques are coarse-grained and do not capture perceptual differences well. For instance, the spatial techniques do not distinguish grayscale vs. color versions. Similarly, color histograms will be identical for a white-black strip vs. an image with black and white dots scattered uniformly. Based on this understanding and insights derived from controlled datasets, we ruled out these approaches.

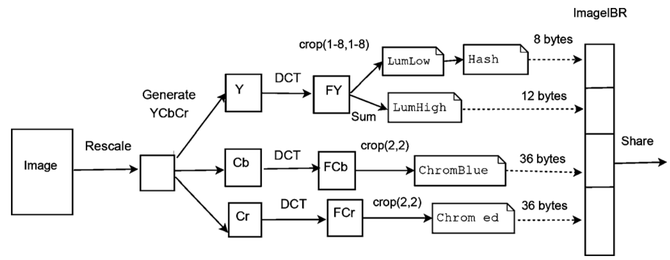

Fig. 3. **Generating image IBRs**. We scale the image to a baseline resolution and convert into the YCbCr representation [12]. For each component in this representation, we generate summaries to create the IBR.

The main intuition behind the frequency domain techniques is that the low-frequency components provide a high-level sketch of the image, and the high-frequency components provide more fine-grained distinctions [17]. Taken together, the resulting fingerprint more closely reflects the underlying information content, making these techniques a good starting point for our system.

Fig. 3 shows how we generate the IBR for an image using frequency domain techniques. We first scale the image to a baseline resolution of 128×128. (128×128 is a good baseline as it is lower than common image resolutions, but high enough to discern detailed structures.) Then, we generate the YCbCr representation of this scaled image [12]. We run the discrete cosine transform (DCT) on the Y, Cb, and Cr matrices to get DCT coefficients.

The IBR consists of two parts: (1) We take the lower end submatrix (rows 1-8, columns 1-8) of Y DCT matrix—which capture more than 95% of the signal energy—and generate a compact 64-bit summary, Hash64, by first finding the median of the coefficients and then quantizing each coefficient to be 0 or 1 depending on whether it is higher or lower than the median [21]. This compact summary reduces the cost of checking if two IBRs are identical. (2) To capture more fine-grained differences, we compute the sum of the high-frequency components of Y, and capture the lower-end 3×3 sub-matrices of Cb and Cr DCT components where most of the signal energy lies. The image IBR is the 92 byte concatenation of the Hash64 and other components (as shown in Fig. 3).

B. Video IBRs

Next, we discuss how to extend the image IBR to video (Fig. 4). One extreme solution is to just think of a video as a sequence of images and simply concatenate the IBR for each frame. However, this makes a video IBR large and expensive to compute. On the other hand, we can use only the image IBRs for the first and last frames. Unfortunately, this allows arbitrary content to be injected between these frames. To ensure tighter binding, it is clear that video IBRs need to encode information about more

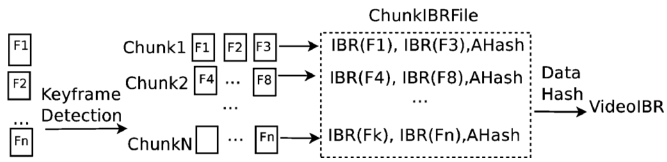

Fig. 4. **Generating video IBRs**. We chunk the video via keyframe detection and compute a per-chunk IBR using the image IBRs of the first/last frames. We apply a traditional data hash to the file containing all the chunk IBRs, and use that hash as the video IBR.

frames. To achieve this, an intuitive option is that we could sample frames on time/byte boundaries; e.g., every 5th second or every N-th frame. However, sampling-based alternatives are not robust as they are sensitive to variations in timing and encoding formats and can result in inconsistent *chunking* across variants.

To ensure tighter binding and consistent chunking, we leverage techniques for scene detection to derive the appropriate chunk boundaries from the information itself [20]. These scene detection schemes identify *keyframes* when the content changes significantly across frames. Intuitively, this is analogous to value sampling in data chunking [28], [35]. To identify the scene changes, we need to select an image feature that is easy to compute and consistent across minor variants. We empirically evaluated various transformations (e.g., format and bitrate changes) to a sample set of 50 movie trailers from Youtube and observed that using the variation in the amplitude of the zero-th frequency of the Y-component yields consistent boundaries across these video transforms. Thus, our chunking algorithm works as follows: Given the zero-th frequency component A_i for each frame i, we compute the distance between two successive frames i and $i+1$, $Dist(i, i+1) = \frac{|A_{i+1} - A_i|}{\min(A_{i+1}, A_i)}$, and check if this crosses a threshold *ChunkThresh*. Each chunk is described by a *chunk IBR* that consists of a two-tuple capturing the image IBRs of the start and end frames $\langle I_{start}, I_{end}\rangle$.

Besides the two-tuple, each *chunk IBR* also conists of a 424-byte audio IBR that we generate using an existing audio fingerprinting algorithm [8]. This scheme first normalizes the audio data into a common format (e.g., mono, 8000 Hz sampled) and generates the frequency domain representation for each frame (roughly 1 sec) of audio. The frequency spectrum for each frame is split into Bark bands (related to human hearing) and a linear regression fit is computed for the power spectra of each band. The coefficients in the linear regressions for the various bands for different audio frames (per second) are packed into a 424-byte fingerprint for each audio chunk. Our experiments with a personal music collection showed that this fingerprint was robust across different types of transforms (e.g., format, quality) (not shown).

Chunk size: A practical issue here is chunk size. Smaller chunks enable more fine-grained adaptation to device and network constraints (e.g., switching to low resolution when battery is low) and provide tighter verifiability (in the limit every frame is a chunk), but also imply more lookup overhead. As a tradeoff between these factors, we set $ChunkThresh = 0.5$ based on controlled experiments (not shown), which yields an average chunk size of $\approx$5 seconds. We also impose a minimum chunk size of 0.5 seconds so that the resolution overhead is low compared to the data transfer time.

Sharing video IBRs: One concern is that an IBR for a long video with many chunks may be too large. For each chunk, the IBR is $\approx$0.6 KB as each image IBR is 92 bytes and the audio IBR is 424 bytes. Thus, for a 20-minute video clip using a 5-second chunk size, the video IBR is roughly 150 KB. Thus, downloading the list of chunk IBRs may increase the video startup delay. We design a practical workaround for this. The IBR that a user posts for a video is analogous to a "torrent" file containing the list of chunk IBRs. (In fact, HTTP chunking based techniques used in video players already do this [37].) That is, in the IBR www.myibr.info?ibr=xyz for the video "xyz" is a data-centric hash of a *manifest file* containing the list of chunk IBRs. Note, however, that the data hash is used only to get the per-chunk IBRs. All subsequent actions—resolution/matching, download, and verification—use the per-chunk IBRs which by design bind to the information.

Using this manifest file containing per-chunk IBRs has two immediate advantages. First, it allows players (or client plugins) to download each chunk IBR in parallel while streaming the video, effectively hiding the latency of downloading the IBRs. Second, it also allows dynamic adaptation on a chunk-level granularity similar to HTTP chunking.

There are other practical benefits of using chunk-level IBR resolution. It can help to find more variants of the video chunks, since full video IBR need not match. In addition, it can accommodate content insertions (e.g., advertisements) in the video. Because of our keyframe detection, ads would be considered as different chunks, and we could still find variants for the chunks of the actual content.

V. IBR RESOLUTION

Having generated IBRs, the next step is to match IBRs in order to resolve the references to the final content that the consumer will view. We begin by discussing the algorithm for matching two IBRs. Then, we describe the design of a scalable resolution service for matching IBRs. We also discuss how we support legacy users and publishers.

A. Matching IBRs

Matching video chunk IBRs largely boils down to matching the constituent image IBRs. Thus, we begin by describing how to match two image IBRs. The Hash64 (Fig. 3) provides a quick check to distinguish two IBRs. However, Hash64 values may differ slightly across different lossy encodings of the same image. To reduce the likelihood of *false negatives* (i.e., the RS is unable to find alternatives even though they exist), the matching process has to accommodate some fuzziness. To this end, we compute the Hamming distance between the two 64-bit values and check if it is within a threshold.

If the Hash64 fields match, we proceed to match the remaining image IBR attributes. Again, these matches are fuzzy; if the differences are smaller than specific thresholds, we classify the images as identical. The use of thresholds in the matching process naturally implies choosing them carefully to control false positives and false negatives. We explain how we tune these thresholds and how they perform on real-world datasets in Section VIII.

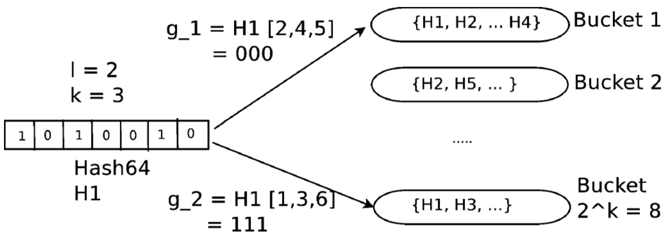

Fig. 5. Bit sampling for LSH with $l = 2$ hash functions and $k = 3$ bits per hash. Here, the hash function g_1 chooses bits at positions 2, 4 and 5, while g_2 picks bits at positions 1, 3, and 6.

B. Locality-Sensitive Hashing

Given the popularity of video content, we expect RSes to process IBR resolution requests at a high rate. The key issue that makes this challenging, as discussed above, is that IBRs generated across different encodings might differ slightly. This is not an artifact of our generation algorithms, but inherent to video content and the lossy encodings and transforms applied to them. Thus, the RS must support *fuzzy matches*.

A naive RS can impose high resolution latencies that impact user experience. For example, we started with an initial MySQL implementation. Given the specific algorithm and thresholds, we used the user defined function (UDF) framework to implement the IBR resolution algorithm, and leveraged existing query capabilities for range/set queries. We populated the databse with roughly 2000 image IBRs and benchmarked the query throughput by averaging over 5000 different queries. We observed that this naive approach could only support ≈150 queries per second even on a high-end server.

Our first step toward scalable fuzzy matching is to use Locality Sensitive Hashing (LSH) [15]. The high-level idea in LSH is to treat the fingerprints as high-dimensional distance vectors and project them to a smaller dimension. To find the nearest neighbors, LSH only compares vectors in this low-dimensional projection, which is significantly cheaper than comparing them in their original representation. The intuition is that similar vectors will (with high probability) be similar in the low-dimensional projection.

Our IBR matching problem is similar at a high-level—we want to find IBRs that are close to a given query. Thus, LSH is a promising starting point. In our setting, we want to compare the Hamming distance between the Hash64 fields. In this case, an efficient way to implement LSH is via *bit sampling* [15], which when applied to matching the Hash64 fields works as follows (Fig. 5). We define l hash functions $g_i, i = 1 \dots l$, where each g_i takes in as input a Hash64 H, selects k random positions from H, and outputs a k-bit vector which is the concatenation of the values at these locations. Then, we map H to the logical *buckets* corresponding to the values of $g_1(H), \dots g_l(H)$. When a query for H' arrives, we retrieve the entries in the buckets $g_1(H'), \dots, g_l(H')$. Then, we compute the exact Hamming distance between H' and each entry in these buckets to identify potential matches. The intuition is that if H and H' are close enough in terms of Hamming distance, they are likely to match when we consider their bits in some k random positions. By choosing l such hash functions, we increase the likelihood of finding such matches.

Extending to video chunk IBRs, we obtain candidate chunk IBRs using an LSH-based check for the first frame of the queried chunk IBR, and directly compare the other IBR fields of these candidates with the input IBR.

There is a tradeoff between two key metrics: the number of entries that need to be processed per input query (and thus the overall throughput) and the false negative rate (i.e., there are candidate IBRs close to a queried IBR, but none of their l hash values match). This tradeoff depends on the choice of l (the number of hash functions) and k (the number of bits per hash function) [15]. Small l and large k can increase throughput, but increase false negative rate. On the other hand, large l and small k can decrease false rate, but at the expense of decreasing throughput. In practice, using a real world video dataset [4], we find that $l = 20$ and $k = 20$ is a reasonable point in the space of tradeoffs. With this setting, we can find close to 90% of the alternatives at a rate of throughput of 8 K queries/second. Note that our overall goal here is to find *some* candidate set of suitable alternatives that meets the users' constraints; we do not need perfect recall.

C. Performance Improvements

Next, we explore further opportunities to improve the throughput, reduce false negative rates and scale the RS.

1) Heuristics: We use three heuristic improvements to improve throughput and reduce false negatives.

- **Pruning:** The first optimization is to stop the search after a sufficient number of matches meeting the consumer-specified constraints are found. This reduces the number of comparisons, thereby improving latency. (To avoid the same matches for all queries and to be "fair" to different hosting sites, we randomize the order of hashtable lookups.)
- **Pre-clustering:** We group nearby IBRs into clusters in an offline pre-processing stage and stored the mapping of each IBR to a cluster. This helps reduce false negatives during lookup. When a matching IBR is found, we retrieve the pre-computed cluster by simply looking up the matching IBR in the stored mapping. We then directly compare all the IBRs in this cluster. Thus, we identify all potential IBR matches (no false negatives), if *at least* one of the IBRs in a cluster has one of the l hash values matching the query IBR. This optimization also improves throughput because we stop the search after finding one such cluster, avoiding further lookups. If the query IBR is already present[4] in the RS, this optimization results in only one lookup.
- **Bypassing LSH:** The last optimization exploits typical viewing patterns where users typically watch a video in sequence. Further, in the common case we expect videos to be whole matches of each other; i.e., if a chunk IBR in a video matches the chunk IBR in another video, it is likely that the subsequent chunk IBRs of the two videos would also match. We exploit this structure to bypass the LSH step. Here, for each chunk IBR, we maintain metadata about its parent manifest file (which contains the list of chunk IBRs) and its offset there. As before, we use a

[4]In general, a query IBR may not be present. For example, an IBR is registered to one RS by a content publisher, but it is queried against another RS maintained by a device vendor

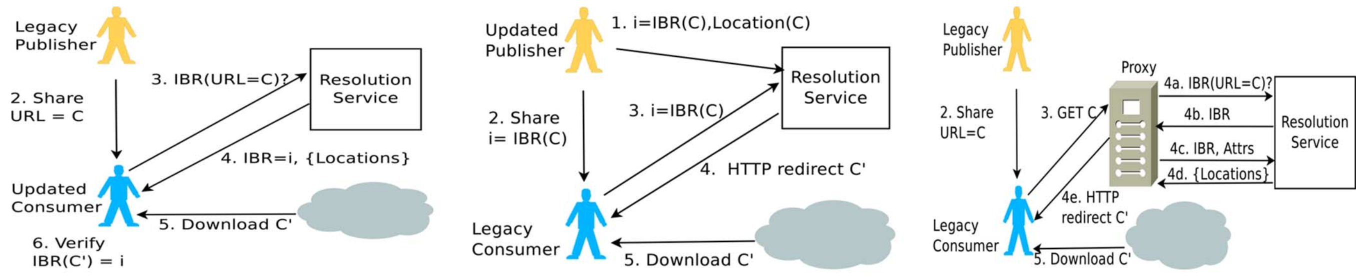

Fig. 6. **Different adoption scenarios for legacy consumers and legacy publishers**. (a) Legacy Publisher, Updated Consumer (LP, UC), (b) Updated Publisher, Legacy Consumer (UP, LC), (c) Legacy Publisher, Legacy Consumer (LP, LC).

LSH-based match for the first chunk. Having identified candidate IBRs for the first chunk, we retrieve their corresponding manifest files. For a subsequent lookup to the chunk at offset j, we fetch the chunks at offset j from these matching chunk-list files, and directly match these IBRs bypassing the LSH stage. If these don't yield sufficient matches (say, because the video is a mashup of scenes from different videos), we fallback to LSH search.

Section VIII-C provides a breakdown and analysis of the scalability improvements from each of these optimizations using real-world datasets. With these in place, the throughput improves to 30–45 K queries/s on a single server.

2) Parallelization: To scale the RS further, we use a simple parallelization strategy. We partition the address space of LSH hashtables (i.e., the g_i logical buckets) across different machines. Thus, different hashtable lookups would be assigned to different machines. As discussed earlier, each query involves lookups from up to l hashtables; we simply randomize the (permutation) order in which we lookup the hashtables for each query. For example, query1 may proceed in the sequence 1-2-5-…, query2 may proceed in sequence 3-1-9-…, and so on. (This is implemented by a lightweight front-end load balancer.) This ensures that the queries are executed in parallel across machines as much as possible.

To put this in context, YouTube serves ≈2 billion videos per day [13]. Assuming an average video length of 5 min and 5 seconds/chunk, this translates to approximately 1 million chunk requests per second. Given that we achieve 45 K queries/sec on a single machine, a cluster of 25 machines can support this YouTube-scale workload of 1 million queries/sec.

D. Legacy Users and Publishers

We now relax the IBR adoption assumptions in the previous section, and ask: (1) Can consumers leverage IBR-based benefits for content shared by legacy publishers who use URLs?, and (2) Can legacy consumers benefit?

To support legacy publishers, the RS provides an additional query interface, where consumers provide direct URLs to the video instead of IBRs; See Fig. 6(a). The RS additionally maintains inverse mappings between URLs and IBR. For such URL-queries, it does an extra lookup to first find the IBR (a simple exact match lookup) and then identifies variants using this IBR.

Two cases arise in supporting legacy consumers: (1) For updated publishers who use IBRs, the RS can use implicit content negotiation for legacy consumers (Fig. 6(b)); e.g., inferring consumer-side characteristics using UserAgent strings or other meta-data in the requests. Alternatively, legacy consumers could set preferences at the RS regarding bandwidth, resolution, formats etc., based on their constraints. When a consumer visits the publisher's page and subsequently contacts the RS, these cookie values are sent to the RS in the HTTP request. The RS issues a traditional HTTP redirect to a suitable alternative. (2) For legacy publishers, the requests from legacy consumers can be sent via an IBR-enabled proxy (Fig. 6(c)). (Otherwise, the consumer will just fetch the URL given by the legacy publisher.)

TABLE III
BENEFITS OF USING IBRS FOR DIFFERENT ADOPTION SCENARIOS. LP/UP REFERS TO LEGACY/UPDATED PUBLISHERS; LC/UC DENOTES LEGACY/UPDATED CONSUMERS

Scenario	Availability	Adaptation	Verifiability of alternates
Today	low	limited	none
LP, LC	high	limited	trust {proxy, RS}
LP, UC	high	medium	trust {RS}
UP, LC	high	limited	trust {RS, server}
UP, UC	high	high	trust {}

The above deployment scenarios vary in the benefits they provide to the consumers (Table III). They clearly differ in availability of suitable alternatives (and hence consumer-specific customization) and support for dynamic adaptation: We get maximum benefits—high availability and high adaptation—when both consumers and publishers are updated to use IBRs. In all cases, using the RS provides higher availability compared to today's URL-based references. In addition, updated consumers can better support dynamic adaptation because legacy consumers are restricted to server-side only mechanisms. For example, the cookie-based customization does not allow dynamic adaptation and users may have to manually change settings to update preferences when their constraints change.

The scenarios also differ in terms of verifiability guarantees, more specifically, the size of the trusted base needed to verify if content received (from a third party hosting site) matches the publisher's intent. When both consumers and publishers use IBRs, the trusted base is minimal. Updated consumers would at most have to trust the RS. Legacy consumers need to implicitly trust both the RS and either the proxy (for legacy publishers) or the content hosting server (otherwise).

VI. VERIFIABILITY

Being bound to the underlying information helps IBRs provide intrinsic verifiability. We discuss the verification step and mechanisms to improve the verifiability further.

Client-side verification: As a video chunk is downloaded, we generate its IBR and match it against the intended chunk IBR. This occurs in parallel as the client is downloading future video chunks. The verification step provides resilence against attacks where an attacker registers a genuine IBR for bogus content. The actual overhead of verification is quite low beyond the cost of decoding the content which the device will incur anyway.

One potential concern is that an attacker can exhaust the client's resources to download, decode, and verify fake content; i.e., the IBR claims to be video X but the actual IBRs of the chunks within the video do not match the IBRs. While this is a concern, we note that this attack's power is bounded. As soon as the client detects a bogus video chunk from a URL, it can stop downloading chunks from that URL and uses alternate sources for the remaining chunk IBRs. Thus, the wasted bandwidth is only for the bogus chunk (5 second worth of bytes). As a further protection against such bandwidth-exhaustion attacks, users can report verification failures to the RS. The RS can revoke these fake mappings (after further checks where it locally computes the IBRs) to avoid using these mappings for future requests.

There are still two possible weaknesses: 1) a determined attacker registers malformed content having the same IBR as some genuine content, and 2) unintentional IBR collisions between content from genuine providers. Next, we discuss mechanisms to overcome these issues.

Access control via scopes: When the publisher registers her IBR and content at the resolution service, she also annotates it with a logical *scope*. These scope annotations allow publishers to constrain the set of third-party providers who can be candidates for serving the content. For example, the publisher may only allow trusted third party providers for high-priority content. (We assume that each provider can be identified; e.g., via a email-id or an identifier in the social network.) For low priority content, it may register a scope with a wildcard to allow anyone to register. When a third-party provider tries to register an IBR-to-URL mapping, the RS checks whether this provider belongs in the given scope, and can accordingly allow or drop this mapping. If the publisher detects misbehavior from some specific provider, she can blacklist this third party and ask RS to remove it from the scope.

Larger or multiple IBRs: In general, using larger IBRs provides tighter binding to the information. For video IBRs, we can add additional components to the per-chunk IBR that capture the variation across frames within a chunk to protect against frame addition or deletion attacks. Instead of choosing a fixed size IBR, publishers can provide two IBRs: (1) a compact IBR used for publishing the references on webpages, and tweets and (2) a larger IBR made available out-of-band (e.g., along with the chunk list file for the video). Consumers can use the larger IBRs for extra verification.

We also envision using multiple complementary IBR algorithms to improve verifiability as different fingerprints capture different characteristics (e.g., spatial gradients vs. colors). In this case, the publisher provides multiple IBRs with annotations to identify the specific algorithm in use, and the consumers verify the IBR for each version. The likelihood of IBR collisions on all versions should be low and thus improve the verifiability guarantees.

Human-assisted reputations: More advanced attackers could generate bogus content that could match the larger IBR as well thwarting the intent of the publisher. To mitigate such attacks, we enhance our framework with out-of-band checks based on ideas from human computation [44].

When the RS receives a request to register a new URL-to-IBR mapping for an existing IBR or a user reports a specific mapping as suspicious, we invoke a reCaptcha-like reputation service [44]. This service downloads the original content (using the original URL that the publisher registered with the RS) and the new/suspicious content and uses votes from users to determine if the new content matches the original content.

The service itself is quite simple. We pick a random frame i from both the original and reported video[5] and users then vote if these are identical/different. Based on these votes, we flag the new video as bogus if more than d frames are voted to be different. For robustness, we implement two well-known safeguards: inserting known results to calibrate voters and collecting a sufficient number of votes per video [44].

One concern here could be the time it takes to discover bogus content. We present a quick analysis that suggests this may not be an issue. Let F denote the number of frames in a video, and let f be the number of frames an attacker has modified. For simplicity, we assume voters do not classify identical frames as different; but may carelessly overlook differences and mark frames that differ only with probability p. The probability of the service detecting a modified frame is $q(f,F) = \frac{f}{F} \times p$, and the expected time to detect *one* modified frame is $\frac{1}{q(f,F)}$. The time to flag a suspicious video is simply the time detect d violations which is $\approx \frac{d}{q(f,F)}$.

With $p = 0.8$ (from our user study in Section VIII-E) and $d = 10$, we need $\approx$100 trials to flag a video with 10% modified content. Given that more than 200 millions reCaptchas are being solved each day [6], our verification service can handle 2 million reported videos or roughly 10% of the number of videos uploaded to YouTube per day [23].

VII. IMPLEMENTATION

IBR Generation: We implement the generation algorithms by extending the `pHash` library [9] involving roughly 2.5 K lines of C++ code. We leverage off-the-shelf audio fingerprinting algorithms [8].

Resolution Service: We built a functional resolution service using a PHP-based web frontend running on top of `Apache` integrated with our optimized LSH-based backend. We use a C++ based LSH backend consisting of roughly 600 lines of code. Our current prototype optimizes three metrics to improve user experience: load time for videos, buffering, and number of "user clicks" required to play the video on the device. Our goal in this paper is not to devise an optimal policy to balance these metrics; rather we provide a mechanism for flexible adaptation. We support both mechanisms outlined in Section III to populate the RS; the RS verifies the binding between content hosted at the URLs and the IBRs before adding IBR-to-URL mappings.

[5]For brevity, assume a one-to-one map between the two videos.

Client-side extension for QoE-aware delivery: One possibility for enabling content negotiation between consumers and the RS is using conventional HTTP `Accept:` headers. However, this option is not *expressive* enough; the existing standard only allows the ability to specify preferences for certain encodings, e.g., `Accept: video/mpg`. Finer grained controls, e.g,. a 3G user wanting `bitrate` $\leq$ 200 kbps, are not possible.

To provide such fine-grained controls, we need client-side modifications. To this end, we developed a Firefox browser extension using Greasemonkey [7], a popular page rewriting tool. The extension processes IBR-ized links on the publisher's site and contacts the RS. It analyzes the available media codecs and plugins, and local conditions such as device type, the type of network interface in use, available bandwidth and battery state, and provides this information to the RS. It rewrites the HTML depending on the RS's response. When the browser loads the new HTML, it simply issues `GET`s to appropriate URLs. For pages with several video references, we batch requests to the RS to avoid multiple round trip delays. Our prototype supports three attributes for content negotiation: device type, format, and bitrate.

The plugin masks the latency of downloading the per-chunk IBRs by fetching them in parallel along with the video stream. When the RS reports multiple alternatives for a chunk, the plugin prefers chunks on the same hosting server and in the same format, unless forced to switch servers because of consistently poor performance. Similarly, when network or device conditions necessitate a change in bitrates, the plugin avoids drastic shifts, choosing instead a feasible bitrate closest to the previously viewed bitrate. The plugin also provides (anonymous) feedback to the RS to track QoE issues specific to a video or hosting site on that device.

Android implementation: We also implemented client-side capabilities for Android (v4.0, Ice Cream Sandwich). In order to provide the IBR functionality in an app-independent fashion, we modify the `MediaPlayer` module, which is part of the webkit middleware. We add the IBR logic to the `setDataSource` function in this module that selects the URL to play. We interpose on this call, and receive an updated URL from the IBR resolver. We use standard APIs to obtain current operating conditions (e.g., `Connectivity Manager` to identify WiFi vs. 3G, `Window- ManagerDisplay` to get resolution, and `BatteryManager` to get the battery status) and report them to the resolver. One concern is that modifying webkit likely requires a device/OS upgrade; we envision this is a feasible option for device vendors and wireless providers who already customize devices.

Proxy: To support legacy clients (Section V-D), we also implemented a simple proxy in C++ that essentially replicates the functions that the client-side browser extension performs while communicating with the RS.

VIII. Evaluation

We address the following questions in our evaluation:

1) *QoE Improvements:* Can IBRs enhance user experience (e.g., load time, buffering)? (Section VIII-A)
2) *Generation:* How do we configure IBRs to ensure low false positives and false negatives? How effective are IBRs in identifying similar/dissimilar content in the wild? (Section VIII-B)
3) *Resolution:* Can the RS resolve IBRs quickly and for a large number of users simultaneously? (Section VIII-C)
4) *Overhead:* Do users perceive delays in viewing pages authored with IBRs? (Section VIII-D)
5) *Verifiability:* How fast can users check IBRs? How practical are the verification guarantees offered by IBRs? (Section VIII-E)

A. QoE Improvements Using IBRs

We begin by showing the quantitative benefits that IBRs provide consumers in realistic settings on actual devices. In particular, we show how IBRs enable network- and device-specific adaptation to improve the user experience.

Bandwidth adaptation: We emulate a blog with an embedded video object in `mpg` format with a bitrate of 1.2 Mbps. We assume that there is an alternative encoding at 620 Kbps, from alternate source. We use a browser with a `VLC` plugin to play the video and a WAN emulator to vary the download speed between 512 Kbps to 2 Mbps.

As a baseline, we consider today's publisher/consumer setup with no IBR support. Here, the publisher uses a video URL that supports single format and resolution (Section II). At low bandwidth (768 Kbps), we observed 6–10 pauses while viewing the video and significant buffering (only 15 s played over a 30 s period). Next, we test the case when the publisher chooses to share the video using IBRs and the client uses our updated browser extension. The video was played smoothly without any buffering induced pauses in this case. At high bandwidth (1.5 Mbps), the browser extension detects and sends this bandwidth information to the RS, which redirects it to the high quality variant. This preliminary result under a controlled setting suggests that IBR-enabled clients can have a better user experience by adapting to dynamic network conditions.

Device adaptation: We emulate an experiment with users sharing video links on social networking sites and consumers viewing those posted links on different devices. For this experiment, we use the same example videos from Table II. We post them on Facebook and use IBRs to share them. In the case of mobile devices without our browser extension, the RS uses implicit content negotiation to redirect clients to the hosting site best suited for the device.

As before, we quantify the end-to-end user experience on the devices w.r.t startup latency for the video to play. Because IBRs redirect users to videos which can play on these devices, it significantly improves content availability; e.g., the third video URL from YouTube does not play on any of the mobile devices, but using IBRs the content can be retrieved from one of the other locations. Furthermore, using IBRs reduces the join time by up to 1 second for Droid on YouTube, up to 4 seconds for Vimeo on iPad2 and up to 19 seconds for DailyMotion on iPhone4 (not shown).

These experiments confirm that IBRs can significantly improve the viewing quality of experience by reducing join time, buffering, and avoiding scenarios where videos were not viewable.

B. Configuring IBRs

We next study how to control the degree of false positives and false negatives introduced when matching different video IBRs. Recall from Section IV-B that a video IBR is essentially a sequence of chunk IBRs, where each chunk IBR has image IBRs of the start and end frames. Thus, we begin by focusing on configuring image IBRs before proceeding to video-specific configurations.

Configuring thresholds for image IBRs: Ideally, we want IBRs that yield zero false positives (i.e., not mark distinct images as same) and zero false negatives (i.e., not mark two identical images as different). We show that it is possible to choose IBR matching thresholds to get close to this ideal. In this process, we are willing to tolerate a small increase in false negatives in favor of completely avoiding false positives (i.e., we are trading off a small decrease in availability for guaranteeing correctness).

As our training set, we used the Univ. Washington image dataset consisting of real-world scenes of nature, people, events, and plants [11]. For each image, we apply the following transforms: (1) change format (from JPEG to BMP and PNG), (2) change resolution (scaled to one-third), (3) change aspect ratio (converting to 1:1), and (4) increase brightness. We chose these specific transforms because we observe that these occur commonly in the wild [4].

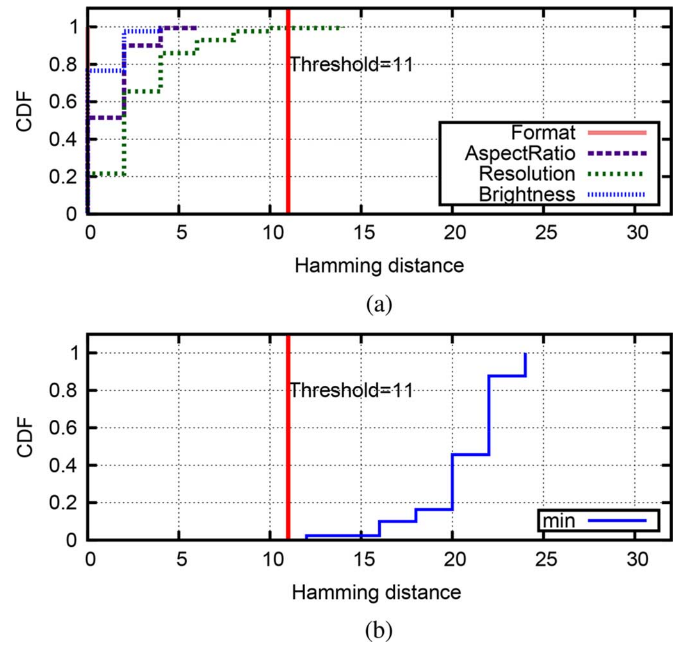

Fig. 7. **Threshold for Hamming distance**. This plot compares the Hamming distances of the Hash64 field across transformed versions of images and distinct images, and shows that we can set a threshold of 11. (a) Across transforms. (b) Across distinct images.

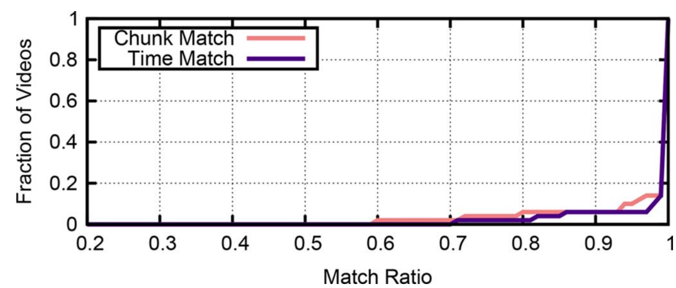

Fig. 8. **Controlled dataset study for videos**. This shows the distribution of match ratio (time/# chunks) across two video formats. The results are similar for variants that differ in the resolution.

Fig. 7(a) shows that the Hamming distance between Hash64 components across transformed variants is at most 15. Fig. 7(b) shows that the *minimum* Hamming distance across different images is at least 13; i.e., for all images, the "nearest" distinct image is at least at a distance of 13. Based on this, we set a threshold of 11 on the Hamming distance between *Hash64* to minimize the false negative rate while maintaining a zero false positive rate. Our dataset is large and diverse and hence the threshold should work for most real world datasets. Moreover, the threshold values can be changed by the provider depending on the deployment. In similar fashion, we use this dataset to select thresholds for other finer grained components of the IBR (details omitted for brevity). With these thresholds, we get a zero false positive rate and a false negative rate of 0.4%. We validated these thresholds on a different set of 250 images from the same dataset and 4223 images from a different dataset [3], and found 1.3% and zero false negative rates respectively and no false positives.

Video IBRs: Having chosen the image IBR thresholds, we move to video IBRs. First, we use a controlled dataset of 50 movie trailers from Youtube to analyze the effect of chunking on the match rate across transformed variants of a video. We apply two transforms: changing the format from `flv` to `avi` and rescaling to 200×180. Fig. 8 shows the distribution of the match ratios across videos with the format change. The result shows match ratio w.r.t time (ratio of total time of matched chunks and total video time length) and number of chunks. We see that format changes have minimal impact; the match rate is $\geq$95% for more than 95% of the videos, both w.r.t time and chunks. Most match misses occur from a known corner case with blank screens where the Hamming distances between similar frames becomes high; we handle this corner case separately, by forcing the chunking algorithm to choose non-blank frames for the first/last chunks, to further reduce false negatives. The result for the resolution change are similar; we do not show this for brevity. To understand how chunking affects match rate, we disable chunking and do a per-frame match and find that the difference between chunk- and frame-level match rates is <0.5% (not shown). This shows that our chunking algorithm yields consistent chunks across video transforms.

Video IBRs in the wild: Moving beyond controlled tests, we run the video IBR algorithms on two larger multimedia datasets collected in the wild: (1) 13, 123 videos fetched using 24 popular queries for a "seed video" (CC Web Video dataset) [4].[6] (2) 120 videos downloaded by querying for a popular movie title across torrent sites (SET dataset) that we manually labeled [35].

We identify two videos as similar only if *all* their chunk IBRs match. Then, we verify this against manually labeled

[6]This is the largest manually labeled video dataset we are aware of.

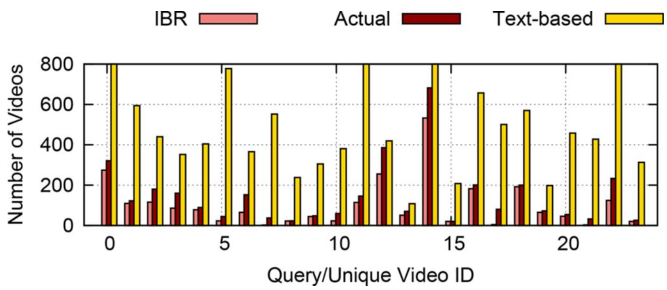

Fig. 9. **IBR effectiveness in the wild**. This compares matches found by IBR with the actual matches on datasets collected in the wild. It also shows the noisy results from text-based queries.

ground truth. Fig. 9 summarizes this result for the CC Web video dataset; Text-based represents the total number of results obtained by the query and Actual represents the number of videos that were manually labeled as similar. The difference between Actual and Text-based indicates the noisy results from text based queries. IBRs did not identify any of the noisy results as similar (i.e., zero false positives). Using the IBRs as-is identifies 72% of the similar videos. Again, the misses were due to the previously mentioned corner case of blank screens. Using the above heuristic to handle this case reduces the false negative rate to less than 5%. Our results for the SET dataset are similar (not shown): zero false positives and roughly 5% false negatives. These results show that IBRs are effective and robust “in the wild”.

C. Resolution Performance

Scalability: We measure the query lookup performance of our LSH-based RS (with and without the optimizations in Section V) using a single core on a Intel(R) Core2 Quad Q6700 2.6 GHz CPU. For this experiment, we use the IBRs for 80,000 chunks from the CC Web Video dataset to populate the RS. For each chunk, we insert half the available alternatives into the RS, and use the other half as query inputs. In our experiments, the entire LSH data structure resides in main memory. In practice, frequenty-looked up entries will be cached in main memory. We used $l = 20$ hash tables and $k = 20$ bits per hash function.

Using the basic LSH structure without any optimizations, we obtain 8 K chunk queries/sec. As discussed in Section V, even this is several orders of magnitude faster than a traditional database backend. Pruning the number of query results to stop after 50 matches are found, improves the throughput to 18.5 K queries/sec. Pre-clustering (without pruning) reduces hash table lookups as the LSH search can be stopped as soon as a cluster is found. In particular, we find that the number of hash table lookups reduces to 9 on average across queries compared to 20 for the basic LSH where all hash tables need to be looked up. For requests that had at least one match, we need only 1.6 lookups on average. Thus, pre-clustering improves throughput to 20 K queries/sec. After combining pre-clustering with the above pruning threshold of 50, the throughput increased to 30 K/sec.

We noted in Section V that we could bypass the LSH step for lookups to subsequent chunks within the same video. The SET dataset had larger clips (15 minutes on average) and several of these were chunked by our algorithm resulting in roughly 80,000 chunks across all video clips. As before, we divide these equally into chunks stored in the mapping and queries. The baseline LSH throughput was 8 K/s that improves to 20 K/s with preclustering. Bypassing the LSH for subsequent (sequential) lookups within the same video, improved the throughput to 35 K queries/sec. When we combined pre-clustering on the first few chunks together with this bypass optimization, the performance improved further to 45 K queries/sec.[7]

TABLE IV
OVERHEAD OF USING IBR-IZED WEBSITES COMPARED TO REGULAR URL VERSIONS FOR THREE WEBSITE TEMPLATES

Template	# Objects	Load time (s)	Increase with IBR (s)
T1	1	0.2	0.1
T2	25	0.6	0.3
T3	50	0.8	0.5

False negatives: As discussed in Section V, the LSH approach could result in false negatives. Using the basic LSH framework on the CC Web Video dataset, we saw a miss rate of 14%. These misses fall in two classes: (1) a queried IBR matches some variants but not others and (2) the query yields no matches although similar variants exist. Fortunately, 96% of all misses are of type (1), and can be addressed via the pre-clustering optimization. We can further address misses of type (2) using one of two extensions: smaller k to reduce the likelihood of misses and using multiple parallel LSH data structures with different random seeds.

D. Increase in Page Load Times

Next, we evaluate the overhead that a user may experience in viewing pages (e.g., blogs) with IBR-ized links due to the need for multiple IBR resolutions. We created three template web pages from blogging and social networking sites, varying in the number of links to multimedia objects. We assume that there is only one version of each embedded object and all objects are hosted at a single remote server. To be conservative in estimating the overhead, all requests go through the RS even though there are no IBR-induced benefits from adaptation. We used a WAN emulator to simulate an average latency of 60 ms between client and the RS.

As Table IV shows, while the IBR-ized version does marginally increase the page load times (mostly because of the additional RTT to contact the RS), the worst case load time is an order of magnitude lower than suggested user tolerance (2 seconds) [25]. This overhead can be reduced even further by optimizing the PHP-based RS or using other web page optimization technique. Finally, it is important to note that this small overhead will be offset by the benefits we saw in Section VIII-A.

E. Verifiability

The previous sections showed the effectiveness and usability of IBRs in normal operating conditions. The final issue we consider is how consumers can verify if the content they view matches the publisher's intent and if they can detect targeted content pollution attacks.

Overhead: The first concern is verification performance. Here, the design of video IBRs facilitates rapid checking: On an

[7]The CC Web video dataset had small videos (<5 chunks); thus we do not report numbers for this experiment.

TABLE V
ATTACKS AGAINST THE IMAGE IBR

Attack	Description	Verifiability?
Inset	embed bogus content	LumLow
Quantization	poor quality/large pixels	ChromBlue,ChromRed
Resize	rescale image, then magnify	LumHigh
Small text insert	random text	none
Replace faces	replace small faces with others	none

Intel Core2 2.66 Ghz machine, it takes 0.02 s to extract the start/end frames of a chunk, 0.2 s to compute the image IBR for each, and 0.01 s to compare against the original chunk IBR. Thus, the total time for verification is 0.25 s per-chunk; this latency can be masked by running these checks in parallel with downloading future chunks. On mobile devices, we expect that this step can be done in hardware as most smartphones and tablets already use hardware-assisted decoding for video [1].

Resilience to pollution: IBRs can easily protect against different content; e.g., against "rickroll"-style attacks in social media. While this eliminates obvious violations, IBRs could be prone to subtle pollution attacks. Next, we study the robustness of our IBRs against such attacks.

Table V summarizes different emulated "attacks". Note that these are subtle attacks; attacks that modify the content substantially will be detected as the Hash64 would differ. We see that some of these attacks are detected via the fine-grained components of the IBR such as the ChromBlue, ChromRed, or the LumLow values (Fig. 3 in Section IV) even if Hash64 fails.

In addition to approaches outlined in Section VI, one approach to tackle such subtle pollution attempts is via a human-assisted reputation service [44].

Video IBRs naturally inherit the verifiability properties of image IBRs. An additional concern with videos are frame insertion (e.g., bogus frames for ads), frame replay, and frame permutation attacks. To address these, we use larger chunk IBRs and add a Hash64 for every frame in the chunk. To compare chunk IBRs, we compute the pairwise Hamming distances between pairs of corresponding frames, and check if the maximum pairwise distance exceeds a threshold. Under all intra-chunk attacks, this value was $\geq$21 confirming the additional robustness offered by larger IBRs.

Effectiveness of User-Assisted Checks: The above results point out that IBR components do provide some level of protection, but they cannot cover all possible attacks. A key rationale for a user-assisted service is to complement the role that IBRs play in verification. To examine its effectiveness, we recruited users to test the feasibility of a human-assisted reputation service. To avoid biasing users, we took two precautions. First, we randomly interspersed identical pairs of frames along with modified pairs. Second, we advertised this as a generic study on "image perception" and did not specify what types of images (frames) or modifications they should expect. As described earlier, users are given a web page which display two images (frames) and they were asked to vote if the images were same or not.

TABLE VI
HOW EFFECTIVE WERE HUMAN VOTERS IN IDENTIFYING DIFFERENT TYPES OF CONTENT POLLUTION ATTACKS

Attack	Detection probability
Replacing faces	0.97
Adding random text	0.82
Blurring frame	0.84

101 unique users (IPs) participated in the study; on average, each user voted on 10 frame pairs.[8] We specifically focus on the attacks from Table V that currently have no intrinsic protection in IBRs: blurring, adding extra text, or replacing small faces. Table VI shows that the observed probability of detecting these attacks is $\geq$0.8. In the context of Section VI, this means that we can set a detection threshold assuming $p \approx 0.8$. We also analyzed if users voted frames that we knew to be identical as different and found that this occurred rarely. These results provide preliminary but promising evidence of the role of a user-assisted service to complement the verifiability guarantees offered by IBRs.

F. Summary of Key Results

Our evaluation shows that the conceptual benefits of IBRs can be realized in practice. Specifically,

- It is possible to configure the thresholds for the IBR generation algorithms to ensure zero false positives and a low false negative ($\leq 5\%$) in the wild.
- A standalone IBR resolution server running on commodity hardware can serve up to 45,000 queries/second with an average latency of 0.05 ms per query.
- Using IBRs can significantly enhance the user experience in social media sharing by reducing 1 "user click" to view content, decreasing the time for videos to load on actual mobile devices by up to 19 seconds, and providing a buffering-free video viewing experience under bandwidth constraints.
- IBRs result in only a small increase in effective page load time for pages with tens of multimedia links.
- We can augment the verification guarantees against more subtle attacks using extended IBR designs and complementing it with a user-assisted service.

IX. RELATED WORK

Our work on IBRs represents the synthesis of ideas both from the multimedia community and the networked systems community. We have already discussed related work in the areas of multimedia fingerprinting and similarity detection in the previous sections. The key difference from this literature is that we envision new use-case for these algorithms that raises new requirements as we discussed earlier. In this section, we focus specifically on prior work in the networking literature that uses similar design principles such as IBRs such as delayed binding and mechanisms to better capture user intent.

Names to capture user intent: There are several prior and parallel efforts for better network and system support to better serve actual user intents. For instance, prior work on Intentional

[8]We were encouraged by the enthusiastic response and persistence of users—some users really did like to click on many pairs.

Naming allows mobile users to specify their "intent" and an in-network resolution mechanism matches it to available services [16]. Similarly, LNA [19] and SFR [32] advocate decoupling identifiers from location and routing analogous to subsequent work on content-centric networking (discussed below). IBRs share their core philosophy that binding protocols to irrelevant details limits flexibility. Our specific focus is on video sharing and decoupling it from content presentation. More recent work on the quFiles system provides file-system level support for seamlessly operating across different user contexts; e.g., low bandwidth, phone vs. desktop. Specifically, quFiles uses context-aware encoding by specialized file names and metadata [43]. Unfortunately, ensuring consistent names and metadata across third-party hosting sites and providers is challenging. Hence, we make a case for an algorithmic basis for IBRs via multimedia fingerprinting techniques.

On a related note, search keywords and queries may appear as a plausible alternative to IBRs that delay the binding between the intent and the actual content delivered. While search engines try to identify similar/related content, it is important to note that IBRs address an orthogonal problem. IBRs complement the discovery to enable flexible and verifiable delivery. That said, search engine providers could offer IBR-like services similar to our proposal for supporting legacy publishers (Section V-D); we are not aware of products that currently offer such capabilities.

Content-centric networking: IBRs share the motivation with recent work in the content-centric networking community [42] that users care about "what" content they consume and not necessariy "where" the content comes from. Thus, they argue for elevating content names as first class principals and also intrinsically bind security in the content names as opposed to server locations. In this sense, IBRs are similar in that we also argue for decoupling names from locations. IBRs go a step further and even decouple the binding between the user intent and the specific formats and encodings by creating higher-level multimedia bindings rather than bit-level bindings. That said, the fuzzy nature of the multimedia bindings raises new challenges with respect to content integrity and resolution that do not appear in these traditional content-centric schenes.

Recent work argues that we do not need significant network upgrade to achieve the security and performance benefits of content-centric networking schemes [26]. Our vision is not in conflict with these arguments. First, we do not make a case for ubiquitous caching; rather we exploit alternatives at third-party sites that exist already. Second, the benefits of IBRs can be achieved in a backwards-compatible fashion without architectural upgrades.

Use of multimedia techniques in networking: Given the dominance of video and image transfers on the Internet there are several other research efforts that have also realized the value of multimedia techniques in the context of network systems. The original vision of IBRs was outlined in an earlier position paper [18]. This work presents a more careful synthesis of the multimedia algorithms, a system for scalable IBR resolution, and an end-to-end implementation. Other recent work has leveraged multimedia algorithms in the context of disaster-recovery applications [45], [41]. These efforts focus on images and largely use the algorithms as deduplication tools. Furthermore, these do not address issues w.r.t. resolution and do not present a full system realization.

X. Conclusions and Future Work

In this paper, we argued the case for IBRs to improve the quality of experience and accessibility of video content, given the growing heterogeneity of device and network operating conditions. The key insight is to bind the content references to the underlying information, ignoring the details of protocols, hosts, filenames, or bits. Consumers can seamlessly choose variants from third-party sites that are the most appropriate fit for their devices and operating constraints, and also verify that the variants match the publisher's intent. IBRs also allow publishers to easily reach a wider audience without significant infrastructure costs. We developed practical algorithms for generating IBRs, a scalable resolution backend, efficient backwards-compatible mechanisms for users to benefit from the power of IBRs, and approaches for providing additional verification.

We believe that our approach of delaying the binding to specific formats and encodings will empower users to express other types of constraints as well and has broader potential to enable new applications. For example, users who need content in their own language or require a much larger resolution (e.g., due to vision impairments) can request that the content meet these requirements. Another use-case might be to adapt the resolution based on the type of network connection; e.g., choosing a low-resolution version on a pay-per-use connection [2]. Similarly, it can also help in challenged networks with resource constraints [45]. We also envision new IBR-enabled caches that identify alternative versions available locally which will be more effective at reducing redundant transfers compared to URL- or data-centric caches.

Acknowledgment

The authors would like to thank Jim Kurose, their CoNext 2013 shepherd, and the anonymous reviewers for their insightful feedback. They would also like to thank Arun Kumar (University of Wisconsin, Madison) for his early contributions to the development of fingerprint algorithms, and Sourabh Yerfule (Stony Brook University) for his help with the Android implementation.

References

[1] "Arm mali graphics processors," [Online]. Available: http://www.pcworld.com/article/2839632/arm-to-bring-4k-to-smartphones-tablets-with-new-gpus.html
[2] "At&t increases data plans for smartphones, tablets," [Online]. Available: http://www.washingtonpost.com/blogs/post-tech/post/atandt-increases-data-plans-for-smartphones-tablets/2012/01/19/gIQAJDIyAQblog.html
[3] "Caltech dataset," [Online]. Available: http://www.vision.caltech.edu/Image Datasets/Caltech256/
[4] CC Web Video: Near Duplicate Web Video Dataset [Online]. Available: http://vireo.cs.cityu.edu.hk/webvideo/
[5] "Cisco forecast," [Online]. Available: http://blogs.cisco.com/sp/comments/cisco_visual_networking_index_forecast_annual_update/
[6] "Google recaptcha," [Online]. Available: http://www.google.com/recaptcha/learnmore
[7] "Greasemonkey," [Online]. Available: https://addons.mozilla.org/en-US/firefox/addon/748/

[8] "The libfooid audio fingerprinting library," [Online]. Available: http://code.google.com/p/libfooid/

[9] "The open source perceptual hash library," [Online]. Available: http://phash.org

[10] "Social recommendations are the number one way us adults discover mobile video," [Online]. Available: http://blog.telly.com/post/51153531136/social-recommendations-are-the-number-one-way-us-adults

[11] "Univ. washington image database," [Online]. Available: http://www.cs.washington.edu/research/imagedatabase/

[12] "The ycbcr color space," [Online]. Available: http://en.wikipedia.org/wiki/YCbCr

[13] "Youtube data," [Online]. Available: http://youtube-global.blogspot.com/2010/05/at-five-years-two-billion-views-per-day.html

[14] YouTube Video Identification Beta [Online]. Available: http://www.youtube.com/t/video_id_about

[15] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," in *Proc. FOCS*, 2006.

[16] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley, "The design and implementation of an intentional naming system," in *SOSP*, 1999.

[17] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transforms," *IEEE Trans. Computers*, vol. 27, no. 1, pp. 291–301, 1974.

[18] A. Anand, A. Akella, V. Sekar, and S. Seshan, "A case for information-bound referencing," in *Proc. HotNets*, 2010.

[19] H. Balakrishnan *et al.*, "A layered naming architecture for the internet," in *Proc. SIGCOMM*, 2004.

[20] J. S. Boreczky and L. A. Rowe, "Comparison of video shot boundary detection techniques," in *SPIE 2664*, 1996.

[21] B. Coskun and B. Sankur, "Robust video hash extraction," in *EUSIPCO*, 2004.

[22] G. DeCandia *et al.*, "Dynamo: Amazon's highly available key-value store," in *Proc. SOSP*, 2007.

[23] Y. Ding, Y. Du, Y. Hu, Z. Liu, L. Wang, K. Ross, and A. Ghose, "Broadcast yourself: Understanding youtube uploaders," in *Proc. IMC*, 2011.

[24] F. Dobrian *et al.*, "Understanding the impact of video quality on user engagement," in *Proc. SIGCOMM*, 2011.

[25] F. Nah, "A study on tolerable waiting time: How long are Web users willing to wait?," *Behaviour & Information Technology*, vol. 23, no. 3, May 2004.

[26] A. Ghodsi *et al.*, "Information-centric networking: Seeing the forest for the trees," in *Proc. HotNets*, 2011.

[27] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proc. VLDB*, 1999.

[28] R. M. Karp and M. O. Rabin, "Efficient randomized pattern-matching algorithms," *IBM J. Res. Dev.*, vol. 31, no. 2, pp. 249–260, 1987.

[29] T. Koponen, M. Chawla, B. Gon-Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica, "A data-oriented (and beyond) network architecture," in *Proc. SIGCOMM*, 2007.

[30] S. S. Krishnan and R. K. Sitaraman, "Video stream quality impacts viewer behavior: Inferring causality using quasi-experimental designs," in *IMC*, 2012.

[31] Y. Liu, F. Li, L. Guo, B. Shen, and S. Chen, "A server's perspective of internet streaming delivery to mobile devices," in *INFOCOM*, 2012, pp. 1332–1340.

[32] M. Walfish, H. Balakrishnan, and S. Shenker, "Untangling the web from DNS," in *Proc. NSDI*, 2004.

[33] C. L. Novak and S. A. Shafer, "Anatomy of a color histogram," in *CVPR*, 1992, pp. 599–605.

[34] A. Oliva and A. Torralba, "Building the gist of a scene: The role of global image features in recognition," *Progress in Brain Research*, 2006.

[35] H. Pucha, D. G. Andersen, and M. Kaminsky, "Exploiting similarity for multi-source downloads using file handprints," in *Proc. of NSDI*, 2007.

[36] T. Rodrigues, F. Benevenuto, M. Cha, K. Gummadi, and V. Almeida, "On word-of-mouth based discovery of the web," in *Proc. IMC*, 2011.

[37] S. Akhshabi, A. Begen, and C. Dovrolis, "An experimental evaluation of rate adaptation algorithms in adaptive streaming over http," in *Proc. MMSys*, 2011.

[38] S. C. Cheung and A. Zakhor, "Estimation of web video multiplicity," in *Proc. SPIE Internet Imaging*, 2000.

[39] L. Shang *et al.*, "Real-time large scale near-duplicate web video retrieval," in *Proc. ACM Multimedia*, 2010.

[40] N. Tolia, M. Kaminsky, D. G. Andersen, and S. Patil, "An architecture for internet data transfer," in *Proc. NSDI*, 2006.

[41] M. Y. S. Uddin *et al.*, "PhotoNet: A similarity-aware picture delivery service for situation awareness," in *Proc. RTSS*.

[42] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, and R. Braynard, "Networking named content," in *Proc. CoNEXT*, 2009.

[43] K. Veeraraghavan, J. Flinn, E. B. Nightingale, and B. Noble, "quFiles: The right file at the right time," in *Proc. FAST*, 2010.

[44] L. von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum, "reCAPTCHA: Human-Based Character Recognition via web security measures," *Science*, 2008.

[45] U. Weinsberg *et al.*, "CARE: Content aware redundancy elimination for disaster communications on damaged networks," in *Proc. HotNets*, 2012.

Ashok Anand received the Ph.D. degree in computer science from the University of Wisconsin, Madison, WI, USA, in 2012.

He is currently a Research Engineer with Instart Logic, Bangalore, India. His research interests include content-aware networking, content-delivery networks, data center networks and cloud computing.

Athula Balachandran is pursuing the Ph.D. degree in the Computer Science Department at Carnegie Mellon University, Pittsburgh, PA, USA, advised by Prof. Srinivasan Seshan and Prof. Vyas Sekar. Her research interests are primarily in the application of machine learning algorithms and data mining techniques towards improving networks and systems.

Aditya Akella received the B.Tech. degree in computer science and engineering from IIT Madras in 2000, and the Ph.D. degree in computer science from Carnegie Mellon University, Pittsburgh, PA, USA, in 2005.

He is an Associate Professor in the Department of Computer Sciences at the University of Wisconsin, Madison, WI, USA. His research spans a variety of topics in computer networking and systems, including software-defined systems, data center networking, video quality of experience, network management, and future network architectures. He has published over 50 papers in leading conferences including SIGCOMM and NSDI, and has served as the program chair for HotNets, IMC and HotSDN. He is a founding PI of the Wisconsin Institute for Software Defined Data Centers in Madison (WISDoM).

Prof. Akella is a recipient of the NSF CAREER award (2008), the NSF Future Internet Architecture Grant (2010), the NetApp Faculty Fellowship (2010), the IBM Ph.D. Fellowship (2003–2005), and several best paper awards (SOCC'13, IMC'10 and COMSNETS'09).

Vyas Sekar received the Bachelor degree in computer science and technology from the Indian Institute of Technology, Madras, India, in 2003, and the Ph.D. degree in computer science from Carnegie Mellon University, Pittsburgh, PA, USA, in 2010.

He is currently an Assistant Professor in the ECE Department at Carnegie Mellon University. Previously, he was an Assistant Professor with Stony Brook University and a Research Scientist with Intel Labs, where he was a member of the Intel Science and Technology Center for Secure Computing.

Dr. Sekar was awarded the President of India Gold Medal at the Indian Institute of Technology, Madras. His work has been recognized with Best Paper Awards at ACM SIGCOMM, ACM CoNext, and ACM Multimedia.

Srinivasan Seshan received the Ph.D. degree in 1995 from the Computer Science Department at University of California, Berkeley, CA, USA.

He is currently a Professor at Carnegie Mellon University's Computer Science Department. He held the Finmeccanica chair from 2004 to 2006. From 1995 to 2000, he was a research staff member at IBM's T.J. Watson Research Center. His primary interests are in the broad areas of network protocols, mobile computing, and distributed network applications. In the past, he has worked on topics such as transport/routing protocols for wireless networks, large-scale network measurements, RAID system design, performance prediction for Internet transfers, ISP multihoming, new approaches to congestion control, large-scale multiplayer games, and large-scale sensor networks. His current work explores the challenges and opportunities related to new networking architectures and cellular, mobile systems. His web page is at http://www.cs.cmu.edu/srini.