Whatever “it” is, you can get it on the Internet[*]: Toward an Information-Based Network Architecture

Aditya Akella, Vyas Sekar and Srinivasan Seshan

1. INTRODUCTION

Recent suggestions for content-centric networking [1, 21, 20], propose naming content directly and to resolve a content name to host locations. Content can then be served from any host holding it.

We take naming and content delivery to the next logical level. We propose an *information-based* network architecture based on naming and querying for *information*. Information is a perceptual entity, a property of content that reflects its defining or most significant features. Multimedia objects that correspond to the same source file but use alternate resolutions, data formats and presentations carry essentially the same information although they differ in their bit encoding.

Our work is motivated by three observations: (1) the dominant fraction of Internet usage is multimedia content; (2) there is great diversity in the availability of such content in terms of sources, presentation formats, and delivery modes; and (3) there is significant heterogeneity among users’ operating conditions, device capabilities, and network performance. Given these facts, we argue that an approach that allows naming and querying for the underlying information can help combine the benefits of two powerful mechanisms: content-based networking (e.g., [1, 21, 20]) and in-network transcoding (e.g., [6, 18]). It enables network end-points to retrieve different versions of content (similar to transcoding) from multiple potential network sources (similar to content-based approaches), giving rise to flexible approaches for client adaptation. More interestingly, it gives rise to new multimedia applications such as information caches and flexible multimedia search.

Two key mechanisms lie at the core of an information-based network architecture: (i) approaches for deriving information fingerprints or *InfoNames*, and (ii) a framework for resolving the information names. The goal of our paper is to conduct a preliminary investigation into these key design issues and explore the viability of an information-based network architecture.

An information-based network requires InfoNames to have several key properties. First, InfoNames must be *presentation-invariant*, meaning that changing the formatting, resolution or presentation of content does not

[*]With due apologies to eBay

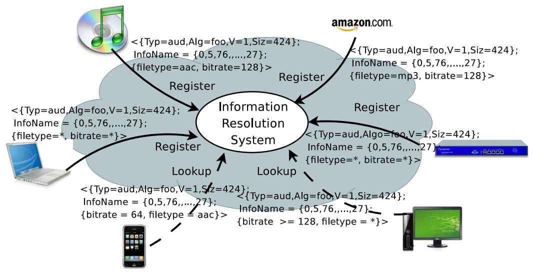
amazon.com
<{Typ=aud,Alg=foo,V=1,Siz=424};
InfoName = {0,5,76,,...,27};
{filetype=aac, bitrate=128}>
<{Typ=aud,Alg=foo,V=1,Siz=424};
InfoName = {0,5,76,,...,27};
{filetype=mp3, bitrate=128}>
Register
Register
<{Typ=aud,Alg=foo,V=1,Siz=424};
InfoName = {0,5,76,,...,27};
{filetype=*, bitrate=*}>
Information Resolution System
Register
Register
<{Typ=aud,Algo=foo,V=1,Siz=424};
InfoName = {0,5,76,,...,27};
{filetype=*, bitrate=*}>
Lookup
Lookup
<{Typ=aud,Alg=foo,V=1,Siz=424};
InfoName = {0,5,76,,...,27};
{bitrate = 64, filetype = aac}>
<{Typ=aud,Alg=foo,V=1,Siz=424};
InfoName = {0,5,76,,...,27};
{bitrate >= 128, filetype = *}>

Figure 1: Example showing how InfoNames can be used. Any network entity can act as a provider; these can be information producers, or peers in the local network, or network middleboxes. Providers *Register* information availability and consumers *Lookup* the information using the InfoName. Users/providers use InfoDescriptors to specify their preferences/capabilities for content formats depending their device and network capabilities.

change its fingerprint, and *bound* to the information. Further, the fingerprints must be *unique* over standard multimedia datasets, and *compact* relative to the size of the content. Fortunately, requisite mechanisms for computing information names with many of the above properties already exist. Applications such as detecting duplicates or copyright violations (e.g., [17, 9]), song identification (e.g., [3]) etc., have driven the development of the necessary robust algorithms for fingerprinting media files. Existing fingerprints do fall short in a few key areas, such as security and integrity of the fingerprints. However, we argue that simple out-of-band mechanisms can be used to address the weaknesses of these fingerprints. These fingerprints can, thus, serve as practical InfoNames in our architecture.

Like past approaches, our architecture exposes a convenient `Lookup()` and `Register()` interface to add and discover data objects (Figure 1). However, unlike past designs, an information based network must also integrate content format negotiation into lookup and registration. To address this need, we envision Lookup and Register messages that carry *InfoDescriptors* that qualify key aspects of both the content (e.g., the bit rate, the creator of content and format) and the InfoName (e.g., the algorithm used to generate the name). Using these descriptors, information providers and consumers can specify their preferences or capabilities for different presentation formats depending on their operating constraints. Since the information-based architecture decouples information from its location (similar in spirit to content-centric networks), it can naturally accommodate diverse data delivery technologies and network intermediaries.

While several pieces of the puzzle are already in place to make information-centric networks viable, there are some open architectural and policy questions that need to be addressed before we can leverage the full potential of this approach. These include issues such as ensuring information integrity, choosing appropriate granularities and structures for naming information, principled approaches for enabling richer lookup and search functionalities, accommodating evolution of InfoName generation algorithms over time etc. Given the potential of an information-centric architecture, these issues merit attention from the networking community and can benefit from active discussions in these directions.

2. BENEFITS OF AN INFORMATION BASED ARCHITECTURE

Transcoding + Content-based networking: Transcoding allows clients to retrieve different versions of the same content from a single network location. Content-centric approaches allow clients to retrieve the same content from different network locations. Generalizing both, our information-based architecture allows clients to retrieve different versions of content from different network locations. Thus, it offers the practical benefits of both approaches: fine-grained adaptation, multi-source downloads, and combinations thereof. For instance, users in wireless and opportunistic environments can query nearby peers for videos using its InfoName. Nearby peers may not have the exact content, but might have alternate versions of it. Thus, users can compensate for unreliability of communication links without needing any support from the original content source.

In addition, our architecture also enables *new services* that would not be possible otherwise:

Opportunities in caching information: We envision "information caches" that index content by their InfoNames. These generalize today's WAN optimizers and offer similar caching benefits, but more importantly, they provide new opportunities. For instance, a infocache can serve low resolution "previews" while the high-resolution version is retrieved across the wide area.

Cross-site multimedia search: Today, searching for videos or video segments across sites is difficult because of different naming/tagging conventions. Our approach can trivially enable such services.

Enriched media services: For example, a collection of peers could host subtitled/translated versions of popular videos. Each video could be hosted at one or more peers, in different formats. Clients can access the enriched video by querying the service using InfoNames along with predicates such as language subtitles.

3. HOW TO NAME "IT"?

Multimedia content currently accounts for the dominant share of network usage and is best suited to benefit from an information-centric architecture. As such, we restrict our discussion of InfoNames to multimedia content, though there is no fundamental limitation in extending our architecture for non-multimedia content such as sensor feeds.

Information and InfoNames: Before we proceed further, we define the following terms informally for clarity. *Information* refers to a "perceptual" entity. The *InfoName* is a "perceptual fingerprint" of this entity. *Content* refers to information coupled with a specific presentation format that can be rendered by higher layer applications. *Data* refers to raw bits with no semantics attached to it. For example, "The Matrix" is "information" with a unique perceptual fingerprint identifying it, `matrix.mpg` is the "content" in MPEG format, and the bit-encoding of the MPEG file is the "data".

The key property of an InfoName is *presentation invariance*. Multimedia content may be in various presentation formats encompassing different resolutions, encodings etc. InfoNames should be invariant across different presentation formats, in order to enable the types of services we envision. For example, the same "image" in `.jpg` and `.png` formats should ideally have the same (or very similar) InfoNames. Similarly, different bitrate encodings of the same audio-visual content should have similar InfoNames.

Information Descriptors: An InfoName in isolation is not useful to applications. We need to provide mechanisms for content negotiation for providers and consumers to decide suitable presentation formats. To this end, we introduce *InfoDescriptors*. An InfoDescriptor is a three-tuple $\langle name\_attrib, InfoName, content\_attrib\rangle$. We explain each of these fields next:

1. Different multimedia types may have different fingerprint generation algorithms and these algorithms themselves may evolve over time. In order to account for different types of media and fingerprinting algorithms, the $name\_attrib$ provides the specification for the InfoName generation algorithm (see specific examples in §3.2), version number for the algorithm, the media type (e.g., image vs. audio vs. video), InfoName size etc. Including the algorithm/versions explicitly allows the architecture to account for and evolve with subsequent developments in fingerprinting algorithms (see §5).
2. The $InfoName$ is a "perceptual hash" that captures the information some content conveys and is

invariant across different presentation formats.

3. For an information-centric architecture to be useful for users and providers, it is necessary to associate content instances and requests with their capabilities. For example, users may want to express interest in only high-quality videos or content in specific formats. Similarly, providers can express their presentation capabilities. The *content_attrib* is a variable length field that carries capability information such as the resolution (e.g., `bitrate = 400kbps`, `size = 800x600`), content creator (e.g., `creator = youtube`, `creator = flickr`), encoding type (e.g., `format = flv`, `format = jpg`) etc.

The use of the InfoDescriptors will become evident when we discuss the information-resolution mechanism in §4. Next, we discuss some key architectural and algorithmic issues involved in the design of InfoNames.

3.1 Architectural Issues

What properties should InfoNames have? The requirements of InfoNames have analogs in the world of *cryptographic hashes* – we want the hash of specific contents to be compact, non-forgeable, easy to compute, data binding etc. We elaborate on these below:

1. **Information Binding:** InfoNames should be tightly *bound* to information. That is, for two contents C_1 and C_2, if $\mathit{Info}(C_1) = \mathit{Info}(C_2)$, then, we want $\mathit{InfoName}(C_1) = \mathit{InfoName}(C_2)$.
2. **Compactness:** InfoNames serve as "keys" to retrieve some information. Thus, they must be much smaller than the actual content they capture.
3. **Ease of computation and checking:** Given some content, it should be easy to generate its InfoName. Also, given a content C and an InfoName f, it should be easy to verify that $f == \mathit{InfoName}(C)$.
4. **Uniqueness:** That is, if $\mathit{Info}(C_1) \neq \mathit{Info}(C_2)$, then, $\mathit{InfoName}(C_1) \neq \mathit{InfoName}(C_2)$ with high probability.[1]

We can trivially ensure uniqueness and compactness by exactly specifying when/where the content was created (e.g., MAC address + timestamp). However, this cannot serve as a useful InfoName as it does not guarantee ease of checking (e.g., we need a global database mapping the information to the name) or information-binding (the same content produced by two sources would have very different InfoNames).

Who generates the InfoNames? The InfoName for a specific content can be generated either by (1) the original information producer; e.g., the music/movie studio that releases the content, (2) other independent content creators; e.g., users who convert files into different formats/encodings for personal use or otherwise, or (3) third-party hosting services; e.g., Youtube or Flickr create InfoNames for videos and images hosted there. At a high-level, we intend the information-centric architecture be relatively agnostic to this choice. The only pertinent issue affecting who generates InfoNames relates to information integrity (see §3.3).

At what granularity should InfoNames be created? We could have a single InfoName for the entire multimedia content or have separate InfoNames for smaller chunks of the content. The choice mainly affects algorithms for retrieval of content. We discuss this further in §5. For simplicity, for the current discussion, we assume that InfoNames are computed over entire content. We do note that our discussion of InfoDescriptors and approaches for deriving InfoNames (next section) hold equally for full content or smaller chunks.

3.2 Algorithmic Issues

Next, we answer the following key algorithmic questions involved in computing InfoNames:

- Are there good algorithms for generating InfoNames for different types of media; i.e., are the names sufficiently distinct for distinct information pieces?
- Are the InfoNames presentation-invariant?
- Do the InfoNames satisfy properties 1-4 from §3.1?

In this regard, we leverage existing techniques from the multimedia community for generating *media fingerprints or signatures*. These techniques have been traditionally motivated by applications such as duplicate detection (e.g., [17]), finding copyright violations (e.g., [9]), song identification (e.g., [3]). Our contribution is not in synthesizing new fingerprinting algorithms. Rather, our key insight is in recognizing that this class of multimedia applications has over the last few years led to a sufficiently mature body of work in robust fingerprinting techniques that can serve as starting points for InfoNames. As a preliminary demonstration of the viability of such fingerprinting algorithms, we consider the case of image and audio content.

We consider a corpus of images[2] and apply the "gist" image fingerprinting algorithm [12]. For each image (a jpg file), we generate 4 transforms – converting it into png and gif formats, and resizing the image by 25% and 50%. Then, for each image and its transformed versions, we compute the "gist" feature vector using the LabelMe toolbox[3] using default recommended settings (4 scales,

[1]"With high probability" appears because of the compactness requirement. If we allow the InfoNames to be arbitrarily long, then we can guarantee perfect uniqueness.

[2]From `http://www.cs.washington.edu/research/imagedatabase/`

[3]`http://labelme.csail.mit.edu/`

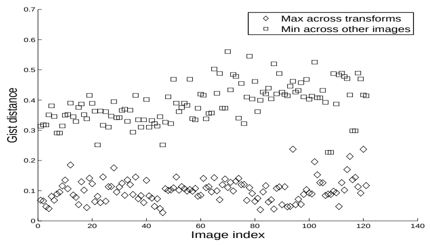

Figure 2: Feasibility of image fingerprinting using the "gist" feature vector

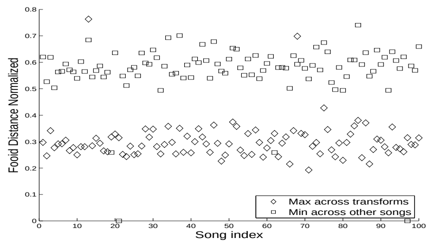

Figure 3: Feasibility of audio fingerprinting using the "libfooid" feature vector

8 orientations per scale, 4 blocks, 4 prefilters). The gist vector for a image is a vector of 512 ($4 \times 8 \times \times 4 \times 4$) float entries, each intuitively capturing the "edge gradient" for each combination of scale, orientation etc. For roughly 800MB of image files, the gists only take up 12MB saved as compressed txt files, meaning that the gists are *compact* taking up only 1.5% of the space as the original image files. Computing each gist takes less than 0.5 seconds using the Matlab library as-is.

Then, we calculate the normalized Euclidean distance between the different gist vectors. Figure 2 compares the maximum distance between the gist vectors corresponding to the same "scene" (i.e., the image and its transforms) against the minimum distance to other "scenes" (other distinct images). The max among the transforms is significantly smaller than the min among other images, confirming two key properties: uniqueness (the fingerprints of each picture are sufficiently distinguishable from the fingerprints of other pictures) and presentation invariance (i.e., fingerprints do not change much across different presentation formats).

Figure 3 shows similar results for audio content with the *libfooid* fingerprint [2].[4] The libfooid algorithm first normalizes the audio data into a lowest common format (e.g., mono, 8000Hz sampled) and then converts this normalized data into its 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 song. This means that the fingerprint is compact: using less than 2MB of space for a 2GB audio collection and easy to compute, around 5 seconds per audio file with an unoptimized implementation using the library as-is. As before, we compute different transforms: converting into ogg format and encoding at bitrates of 96 and 64 Kbps. Again, the fingerprints are both unique and presentation-invariant.

Similarly, there are several candidate fingerprinting algorithms for video content. Some of these extend image fingerprinting. For example, we can apply the gist to each "key frame" in the video, and simply use a collection of these gists (e.g., [10, 8]) as InfoNames. Other approaches leverage temporal information and use changes across scenes as fingerprints (e.g., [9, 11]).

[4]Over a personal music collection.

3.3 Information Integrity

Content-centric approaches, where cryptographic content hashes are used to name content, can guarantee *data integrity*. A user can verify that the data it downloads from someone is the data intended by the original content producer by checking the data's hash against that of the original content. A natural question is if we can analogously guarantee *information integrity*: if a user downloads some content from a third party and the information fingerprint matches the fingerprint indicated by the information producer, does this guarantee that the user was not served bogus or malicious content? [5]

The fingerprinting mechanisms that we have borrowed from the content identification literature (§3.2) are robust to *evasion*; i.e., preventing adversaries from forcing the fingerprints of conceptually identical content to be different. For integrity, however, we need robustness to *pollution*; i.e., preventing adversaries from generating fake content (e.g., malware, "rickrolling") with a specific InfoName. It is unclear if the current approaches are robust against pollution. Designing such robust approaches is subject for future work.

However, we can provide information integrity while using existing fingerprinting approaches by leveraging out-of-band mechanisms. Some possibilities are:

- **Trusted Sources:** We can approximate information integrity by trusting some content sources, e.g., large providers or trusted peers, or trusting InfoNames created by certain sources.
- **Reputation Mechanisms:** Even if we don't trust the sources, we can use reputation systems like Credence [22] to provide some degree of integrity.
- **Attestations from trusted sources:** The information producer or a third-party can certify and bind specific content to its information fingerprint. This

[5]Integrity differs from uniqueness (§3.1), where the focus is on the common case rather than the adversarial case in the case of integrity. For uniqueness, we desire that the InfoNames of randomly chosen content segments do not collide.

attestation can be added to the $content\_attrib$ field; subsequently we only need to check for content integrity to ensure information integrity.

4. A FRAMEWORK FOR INFORMATION RESOLUTION

API for Information Resolution: The primary goal of the API is to allow information providers and consumers to publish and receive information in suitable content presentation formats. If we think of InfoNames as a logical extension of the data-centric architectures, this means that users (providers) not only specify that they desire (serve) some specific information, but also specify content presentation *capabilities* (e.g., formats, resolutions, bandwidth, server location, choice of server etc). This is similar to the notion of content negotiation (e.g., [19]).

We define two main functions in the InfoNames API: *Register* and *Lookup*. As the names suggest, Register is used by an information provider (either the information producer or a third party provider), and Lookup is used by consumers. In addition, the API also provides an *GetInfoNames* function to bootstrap clients with the set of InfoNames (i.e., fingerprints) constituting an object.

Query Resolution: Both Lookup and Register take an *InfoDescriptor* as the input. Recall that an InfoDescriptor carries both the InfoName and additional $content\_attribs$ describing content capabilities. For a specific content instance, each content_attrib entry in an InfoDescriptor is a single key-value pair. However, in the Register and Lookup functions, we generalize the entries in the content_attrib to be *ranges* or *sets* to specify a range or set of available/desirable content presentations respectively. For example, a provider who can serve a video at different formats (or alternatively provide transcoding services) issues Registers with the content_attrib containing the set of offered formats.

The resolution mechanism can leverage existing techniques such as hierarchical resolution (e.g., DNS), DHT-based publish-subscribe [13] or simply use scope-limited broadcasts, or any combination thereof. The choice of the resolution mechanism depends on the operating environment and we do not explicitly commit to one mechanism over another.

In addition to matching InfoNames between Lookup and Register messages, the resolution mechanism also matches the content_attrib fields, implicitly serving as a content negotiation step. For example, consider a cellular user with a slow network connection of 200kbps interested in a specific video with InfoName f. This user can issue an Lookup for f with the InfoDescriptor containing an entry for `bitrate` $\leq$ 200kbps, and the resolution mechanism only returns videos that have a bitrate lower than 200kbps.

The resolution system may have additional flexibility to perform richer matches on the InfoNames or match some content_attribs optionally. We discuss this next.

Enhanced search capabilities: Since InfoNames are based upon the information content of an object, they raise the possibility of designs in which similar objects have similar names. This, in turn, may enable the InfoNames architecture to support the lookup of objects based on similarity. For example, users can search for similar or related music or videos (e.g., `www.pandora.com`), if the semantics of the InfoNames allow such queries. There are a few challenges to address before such searches are possible. Different fingerprinting algorithms may vary in how they measure similarity; e.g., some techniques may ensure that similar content shares a common prefix, others may indicate similarity in the number of bits flipped. Each such similarity measure requires a different mechanism for discovering objects with similar names. For example, a range lookup mechanism might be applicable for some techniques but not others. One possibility is to have different lookup techniques associated with different naming algorithms.

In-Network vs. Host-based Query Result Filtering: Depending on the operating environment, the resolution system can return unfiltered results without matching the content_attribs, or filter the results inside the network and only return a subset of the matches. This behavior may also be specified as part of the lookup protocol. One of the key tradeoffs is the burden these choices impose on the infrastructure and end devices. Requesting all results, especially for widely replicated items, can create high overhead both within the lookup system and on end-user devices (e.g., poorly connected mobile devices that may be overwhelmed by the responses). However, providing incomplete results may make it difficult for end-points to make informed decisions. We envision a flexible resolution system that can accommodate multiple modes suitable for different scenarios.

5. OTHER ARCHITECTURAL ISSUES

Granularity of InfoNames: Different granularities will enable or restrict the choice of data retrieval algorithms. For example, if an entire video has a single InfoName, then users have to download it from a single source, but if the video is broken to multiple segments each with an InfoName, users can download from multiple sources. An information-centric architecture will be most useful when different network entities agree on information-chunking mechanisms with *consistent* semantics. If different entities chunk at arbitrary granularities, the benefits of chunking will be lost. One possibility is using techniques similar to value-sampling (e.g., [14, 16]), where chunk boundaries are determined from the data itself. That is, we identify information-chunk boundaries if the InfoName matches a predefined pattern. This ensures that chunks from different versions of the same content have consistent information (e.g., first 3 minutes of a movie). However, computing this efficiently may require that we compute InfoNames incrementally.

Structured vs. Unstructured Names: A closely related issue to naming granularity is whether InfoNames should be "flat" (have no structure) or should have some structure (e.g., hierarchy). For example, if we segment a 1 hr video into 5 minute segments, should the InfoName for each 5-minute video segment also carry the fingerprint of the original 1-hour segment to convey that this conceptually belongs to the larger video? One approach is to augment the $content\_attrib$ with entries to carry such structural information.

Changing InfoNames over time: It is likely that the algorithms used to generate InfoNames will change over time, as new developments arise in the multimedia community. Note that the name_attrib associated with an InfoDescriptor specifies the algorithm used to generate a name, which makes it easy to support multiple fingerprinting techniques. While our architecture has the requisite mechanisms to support such evolution (e.g., we can have two different InfoDescriptors and add content_attrib entries for backward compatibility), there are some open policy questions such as: How and when are new InfoNames computed? To what degree should the system support translation between names? Does data have to be re-chunked to reflect new naming algorithms (e.g., if we use value-sampling approaches) ?

6. RELATED WORK

Naming Architectures: In Intentional Naming for mobile networks [4], users specify their "intent" and an in-network resolution mechanism matches this with available services. At a high-level, InfoNames too convey "intent", but specifically targets presentation-agnostic information lookup rather than resource discovery. LNA [7] and other proposals advocate decoupling identifiers from location and routing. We share their core philosophy that binding protocols to irrelevant details unnecessarily limits flexibility. Our specific focus is on decoupling information from content-presentation.

Data-centric networking: Data-centric architectures [21, 20, 1] are motivated by the fact that users care about content and not who is serving it. We take this stance one step further: users care about "information" and can be oblivious to both location and content presentation. Data-centric approaches are particularly appealing for challenged environments [5, 15]. Our architecture applies to such settings as well.

Network transcoding and service composition: Our work shares the same motivation for dynamic service adaptation as in-network transcoding (e.g., [6, 18]). These assume the existence of proxies for on-demand transcoding that are application- and client-aware. Our proposal inherently provides these benefits and can additionally can accommodate such proxies.

7. CONCLUSIONS

Our proposal for an information-centric network architecture represents the synergistic combination of developments in content fingerprinting in the multimedia community with the architectural principles underlying content-centric networks, layered naming, and in-network transcoding. That is, this generalizes the flexibility, performance, availability, and adaptation benefits of these diverse domains. At the same time, our work can enable new richer services that cannot be provided by each of these solutions in isolation. In these respects, we believe that there are broader architectural implications of developing an information-centric network infrastructure, that merit attention and would benefit from active discussions in the networking community.

8. REFERENCES

[1] A Conversation with Van Jacobson: Making the case for content-centric networking. ACM Queue, Feb 2009.
[2] libFooID - free fingerprinting library. `http://www.foosic.org/`.
[3] Shazam. `http://www.shazam.com`.
[4] W. Adjie-Winoto, E. Schwartz, H. Balakrishnan, and J. Lilley. The design and implementation of an intentional naming system. In *Proc. of SOSP*, 1999.
[5] K. Fall. A delay-tolerant network architecture for challenged internets. In *Proc SIGCOMM*.
[6] A. Fox, S. D. Gribble, E. A. Brewer, and E. Amir. Adapting to network and client variability via on-demand dynamic distillation. In *Proc. ASPLOS*, 1996.
[7] H. Balakrishnan et al. A layered naming architecture for the internet. In *Proc. SIGCOMM*, 2004.
[8] A. Hampapur and R. M. Bolle. Feature based indexing for media tracking. In *Proc. of ICME*, 2000.
[9] P. Indyk and N. Shivakumar. Finding pirated video sequences on the internet. Stanford Infolab Technical Report, Feb. 1999.
[10] A. K. Jain, A. Vailaya, and X. Wei. Query by video clip. *Springer-Verlag Multimedia Systems*, 1999.
[11] M. Y. M. Naphade and B.-L. Yeo. A novel scheme for fast and efficient video sequence matching using compact signatures. In *Proc. SPIE, Storage and Retrieval for Media Databases*.
[12] A. Oliva and A. Torralba. Building the gist of a scene: the role of global image features in recognition. *Progress in brain research*.
[13] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys*, 35.
[14] H. Pucha, D. G. Andersen, and M. Kaminsky. Exploiting similarity for multi-source downloads using file handprints. In *Proc. of NSDI*, 2007.
[15] R. Krishan et al.,. The spindle disruption-tolerant networking system. In *Proc MILCOM*.
[16] M. Rabin. Fingerprinting by random polynomials. Technical report, Harvard University, 1981. Technical Report, TR-15-81.
[17] S.-C. Cheung and A. Zakhor. Estimation of web video multiplicity. In *Proc. SPIE Internet Imaging*, 2000.
[18] S. Gribble et al. The Ninja Architecture for Robust Internet-Scale Systems and Services. *Computer Networks, Special Issue on Pervasive Computing*, 35(4), Mar. 2001.
[19] S. Seshan, M. Stemm, and R. H. Katz. Benefits of transparent content negotiation in http. In *IEEE Global Internet Symposium*.
[20] T. Koponen et al. A Data-Oriented (and Beyond) Network Architecture. In *Proc. SIGCOMM*, 2007.
[21] N. Tolia, M. Kaminsky, D. G. Andersen, and S. Patil. An architecture for Internet data transfer. In *Proc. NSDI*, 2006.
[22] K. Walsh and E. G. Sirer. Experience With A Distributed Object Reputation System for Peer-to-Peer Filesharing. In *Proc. NSDI*, 2006.