SmartEye: Real-time and Efficient Cloud Image Sharing for Disaster Environments

Yu Hua* Wenbo He† Xue Liu† Dan Feng*
*Wuhan National Lab for Optoelectronics, School of Computer
Huazhong University of Science and Technology
Wuhan, China
{csyhua, dfeng}@hust.edu.cn
†School of Computer Science
McGill University
Montreal, Canada
{wenbohe, xueliu}@cs.mcgill.ca

***Abstract*—Rapid disaster relief is important to save human lives and reduce property loss. With the wide use of smartphones and their ubiquitous easy access to the Internet, sharing and uploading images to the cloud via smartphones offer a nontrivial opportunity to provide information of disaster zones. However, due to limited available bandwidth and energy, smartphone-based crowdsourcing fails to support the real-time data analytics. The key to efficiently and timely share and analyze the images is to determine the value/worth of the images based on their significance and redundancy, and only upload those valuable and unique images. In this paper, we propose a near-real-time and cost-efficient scheme, called SmartEye, in the cloud-assisted disaster environment. The idea behind SmartEye is to implement QoS-aware in-network deduplication over DiffServ in the software-defined networks (SDN). Due to the ease of use, simplicity and scalability, DiffServ supports the in-network deduplication to meet the needs of differentiated QoS. SmartEye aggregates the flows with similar features via a semantic hashing, and provides communication services for the aggregated, not a single, flow. To achieve these goals, we leverage two main optimization schemes, including semantic hashing and space-efficient filters. Efficient image sharing is helpful to disaster detection and scene recognition. To demonstrate the feasibility of SmartEye, we conduct two real-world case studies in which the loss in Typhoon Haiyan (2013) and Hurricane Sandy (2012) can be identified in a timely fashion by analyzing massive data consisting of more than 22 million images using our SmartEye system. Extensive experimental results illustrate that SmartEye is efficient and effective to achieve real-time analytics in disasters.**

I. Introduction

Natural disasters often incur severe loss of human lives and properties. Smartphone based crowdsourcing for sharing images has taken important role in disaster relief. For example, in Hurricane Sandy (2012), 800,000 pictures from professionals and amateurs were uploaded to Instagram. In Typhoon Haiyan (2013), as part of the relief efforts, social media was mined by volunteers to tell where the most help is needed [1]. Hence, cloud image sharing is important and helpful to disaster detection, scene recognition, and loss query in the disaster environments.

Millions of people use smartphones with cameras and new photographic technologies are emerging that allow individuals to constantly record their lives. According to IDC reports [2], multimedia represents 68% of all Internet data in 2012. It is estimated that nearly 50% of cloud based services will be the contents in the form of images by 2020. Nowadays, users routinely upload or share more than hundreds of Gigabytes of images on the cloud. Hence, there is a growing potential for users to share the pictures taken in disasters, and the crowdsourcing aid system generally contains massive crowd generated images [3]. However, the huge amount of image data in the crowdsourcing aid system incurs nontrivial challenges to share, search and mine images in a real-time manner.

Existing crowdsourcing systems usually rely on humans to tag or annotate images, such as Google Image Labeler, Amazon Mechanical Turk and CrowdSearch, which incurs long processing latency. To accelerate the image data processing, previous efforts have been made using multimedia based tools to analyze a massive amount of crowd images [4], [5]. However, these tools consume not only substantial energy but also large fraction of bandwidth. This is because the current crowdsourcing systems overlook the near duplicates among the shared image data. Consequently, important resources (e.g., computation, storage and network bandwidth) and time may be wasted on delivery to process the redundant information. For example, by examining several common applications such as web browsing, maps, application install, email, and Facebook, the results show that redundant storage does indeed affect the overall performance in smartphones [6]. Therefore, it is of paramount importance to the support for real-time sharing and searching images, as well as efficiently dealing with three main challenges.

Slow Processing Speed. Existing deduplication schemes can handle either content-aware images or system-aware chunks based on the exact-matching methodology [7]. In order to guarantee the accuracy, this methodology requires a large amount of auxiliary metadata. A 1MB-size image may produce 200KB-size features via PCA-SIFT [8]. When dealing with millions of images or even larger, the heavy space overhead will exceed the memory size and cause frequent access to the slow-speed hard disks. For example, a query on the 2-million-image set require 12.6 minutes [4], [9]. Existing schemes execute the deduplication operations in either source or destination systems, which cause long processing latency. The slow processing speed hence results in potential data staleness.

Substantial Energy Consumption. Massive images are generated by the smartphones of users. Many users have to charge their smartphones after a single day of moderate usage. For example, for smartphone dislikes, ChangeWave conducted a market study in 2011. 38% of the respondents reported that their biggest complaint was the battery life [10]. In general, image sharing and communication with the cloud can consume a large fraction of the limited energy. If we reduce the

978-1-4799-8381-0/15/$31.00 ©2015 IEEE

number of transmitted images, substantial energy can be saved. Moreover, a fast operation response is also helpful to save energy due to the decrease of waiting time in the smartphones.

Potential Bandwidth Bottleneck. The crowd-generated data generally have the temporal and spatial locality due to clustering property. For example, when reporting and sharing a hot-spot and interesting event, most of the images uploaded by the crowd are similar and will be simultaneously uploaded to the cloud servers. The concurrent operations from many users possibly result in the transient bandwidth bottleneck. Users hence obtain very slow operation response. Moreover, due to the potential bandwidth bottleneck, frequent attempts to connection with the network in smartphones may further aggravate the bandwidth shortage and consume more energy.

The crowd can generate a large amount of similar or duplicate multimedia images to guarantee the image quality. How to handle the exponential increase of such duplicate and similar images has become an important problem to commercial sites, such as Imagery, Google, Yahoo!, Bing Images search, Picsearch, Ask Images Search, etc. In the disaster relief, high-resolution smartphones offer high image quality and multiple angles, but incur heavily redundant images. Although repeatedly taking pictures from the same or neighboring people can further guarantee the quality of snapshots, it will definitely produce more redundant images from a user's view. The redundant images, if all are uploaded, require lots of bandwidth and consume substantial energy of smartphones. In the meantime, due to the heterogeneity of uncoordinated user devices, it is difficult to accurately identify and confirm the similar images to be uploaded. Conventional approaches need to communicate with the destination cloud systems to determine which images are similar via similarity indexing and matching, thus incurring long latency.

To address these problems, we propose a novel QoS-aware DiffServ-over-SDN scheme, called SmartEye, with a design goal of in-network coarse-grained deduplication. SmartEye can obtain energy savings and improve bandwidth efficiency. The idea behind SmartEye is to aggregate the images with similar features to the same class and contain the same label in the DiffServ domain. Due to aggregating similar images, they hence have large probability to be deduplicated within the network. Two main important problems are: 1): *How to accurately and efficiently aggregate similar images?* 2): *How to label similar images to support in-network deduplication?* We explore and exploit the similarity property within and among image sets via semantic-aware hashing [11] and filters [12]. The compactness makes SmartEye attractive for mobile image applications, since they can be communicated from the phone to a remote search engine server at extremely low energy cost. Note that we do not remove any near-duplicate images, but upload the valuable and unique ones, due to the bandwidth and energy constraints. The remaining images can be uploaded when the bandwidth and energy are sufficient. Specifically, we make the following contributions.

(1) In-network Deduplication Design. In order to improve the network transmission performance, we propose the in-network deduplication approach, in which the deduplication operations can be completed in the intermediate nodes, rather than the conventional source or destination end systems. The benefits in the in-network deduplication are to reduce the transmission latency and decrease the bandwidth requirement. We identify the similar images via the low-complexity semantic hashing. More importantly, the in-network deduplication efficiently decreases the complexity and overheads in the core nodes.

(2) QoS-aware DiffServ Architecture. In order to implement the in-network deduplication scheme, SmartEye makes use of the QoS-aware DiffServ model, in which the similar data are aggregated into the same flow and obtain the identical transmission quality. We use the features to represent images and group these images via Locality-Sensitive Hashing (LSH) [11], [13]. The images in a group can be mapped into the same DiffServ aggregated flows. Since the intermediate nodes contain the frequently accessed data, the aggregated flows with high redundancy can efficiently find the redundant data with a high probability.

(3) Real Implementation. We implement all components and functionalities in a cloud-assisted prototype system. We collect two real-world image sets that consist of more than 22 million images (over 60TB storage capacity). Using these real-world image datasets as case studies, we evaluate the performance of SmartEye and compare it with the state-of-the-art schemes. The case study evaluation demonstrates the efficiency and efficacy of SmartEye.

The rest of this paper is organized as follows. Section II presents problem statement. Sections III and IV respectively describe the SmartEye architecture and implementation details. We evaluate the performance in Section V. Section VI presents the related work. Section VII concludes the paper.

II. Problem Statement

Our research concentrates on real-time and cost-effective data communications in the cloud-assisted systems. The main function is to fast identify similar images from the massive image datasets and transmit representative and unique, rather than all, images to obtain bandwidth and energy savings.

In order to address these problems and achieve the design goals, we need to accurately identify and aggregate similar images, and execute the mapping between the labels and in-network deduplication operations. This is more important in the disaster relief, in which the bandwidth is limited and not reliable, and the smartphones with limited energy are difficult to be re-charged due to the infrastructure destruction. The problem stated is transformed into how to fast identify correlated and similar (near-duplicate) images from a big image dataset. Near-duplicate images can be defined by the requirements from given applications, such as personal image collections, photo sharing websites and social networks. In the conventional system-level deduplication, two files are considered duplicates if their bit-streams have an exact match. In practice, duplicate images include not only identical copies but also slightly "photoshopped" copies of an original image source via digital photometric or geometric transformations. These transformed images are based on the same original content, but present themselves as distinct files from a file system's viewpoint, often with little or no detectable redundancy from a conventional system-level deduplication's viewpoint.

The definition of similarity is important to the efficiency and efficacy of SmartEye. Content-based similarity detection

allows images of the same scene but with slightly different viewpoints and illuminations to be considered similar. Therefore, *the definition of near-duplicate images is associated with not only what photometric and geometric variations are deemed acceptable, but also what a user's perceptual understanding is subjective* [14].

III. The SmartEye Design

In this section, we present the architectural overview of SmartEye. This is followed by descriptions of the clients and servers, as well as the workflow of SmartEye.

A. The Architecture

Conventional data deduplication is executed in either source clients or destination cloud servers. Specifically, the source-based deduplication can delete the redundancy before sending the data to the cloud servers. This scheme allows client software to communicate with the server, in which arriving data are compared with previously stored data to determine the data to be uploaded. The source deduplication often requires frequent communication with the servers and causes relatively long indexing latency in the servers. Moreover, in the destination deduplication, all data need to be uploaded to the servers before executing the deduplication operations in the servers. A large fraction of bandwidth is consumed to transmit the original data.

We argue that both source and destination schemes have to handle the data in an end-to-end manner. These schemes can obtain the operation response only after carrying out end-to-end communication, thus failing to deliver high performance in the cloud. Software defined network (SDN) offers an opportunity to support an in-network deduplication, in which we can execute in-network deduplication within the network [15]. Figure 1 shows the SmartEye architecture that consists of three main components, including in-network deduplication, QoS-aware DiffServ and SDN infrastructure.

The SDN infrastructure provides efficient network communication services. *The SDN system is similar to the hierarchical computer system that mainly consists of in-memory cache and hard disk in terms of managing data.* Specifically, the controller, like a hard disk, maintains the full data index and decides the routing path. The SDN switch like a memory cache maintains the frequently-used data and supports local fast deduplication. Moreover, the controller stores the index of the data that have been transmitted though the end-to-end SDN. The full-mirror knowledge is helpful to the controller to efficiently compute and decide the routing. In order to alleviate the overhead of the controller, we design the local cache in the SDN switch that leverages simple LRU caching policy to maintain and manage the "hot-spot" data. In this way, the switch removes duplicate data and groups near-duplicate data into the aggregated flows without the needs of communicating with the controller.

The QoS-aware DiffServ offers coarse-grained and class-based scheme for differentiated communication services that result in heterogeneous service quality. For example, DiffServ allows important flows to be low-latency and others to use simple best-effort service. To differentiate the service quality, DiffServ supports packet classification via a 6-bit differentiated

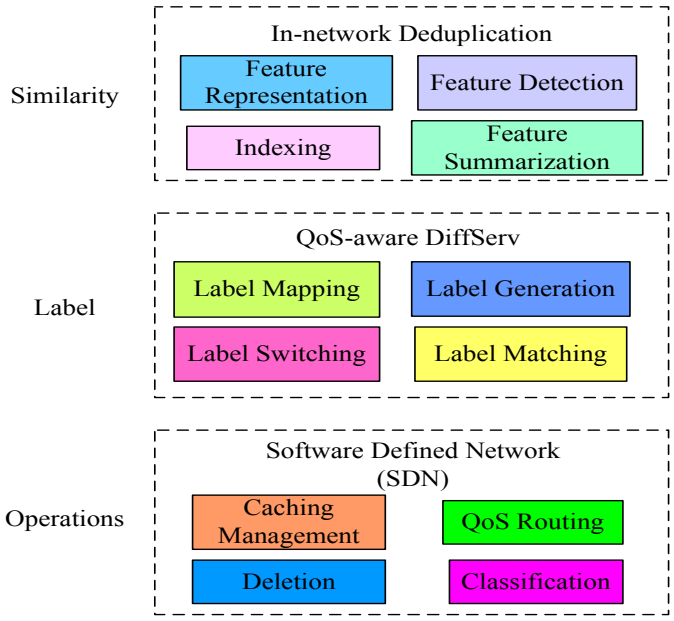


Fig. 1. The DiffServ over SDN architecture to support in-network deduplication in SmartEye.

services code point (DSCP), which exists in the 8-bit Differentiated services Field (DS field) in the IP header. Moreover, in order to specify a concrete class, DiffServ needs to classify and mark packets. In the DiffServ-aware domain, the quality of service is tightly related with the labels that are mapped to per-hop behaviors (PHBs). SmartEye makes use of the PHBs to exhibit the in-network deduplication operations in the intermediate switches that contain the frequently visited data. SmartEye allocates data packets to a limited number of classes. Based on these classes, DiffServ handles the transmitted data in a differentiated manner.

The in-network deduplication component can implement the deduplication within the network, rather than conventional source or destination end systems. SmartEye identifies similar images by using LSH that uses a family of locality sensitive hash functions. In the multi-dimensional feature space, these functions consist of linear projection over random directions. The rationale of LSH design is that close-by data points have higher probability of being hashed into the same or close-by buckets. With the aid of similar features, SmartEye can efficiently identify and deduplicate data redundancy.

B. Network Communications

SmartEye supports an efficient and cost-effective scheme for near real-time data analytics for fast sharing images. It uses a simple and easy-to-use index structure to allow the indexing operations to obtain in-memory performance and decrease the complexity.

Figure 2 shows the model of network communications between clients and servers in SmartEye, which consists of their main components and modules. SmartEye consists of the main functional modules of lightweight feature extraction and space-efficient summarized vectors in clients, and semantic-aware grouping and caching in the servers. All these components can

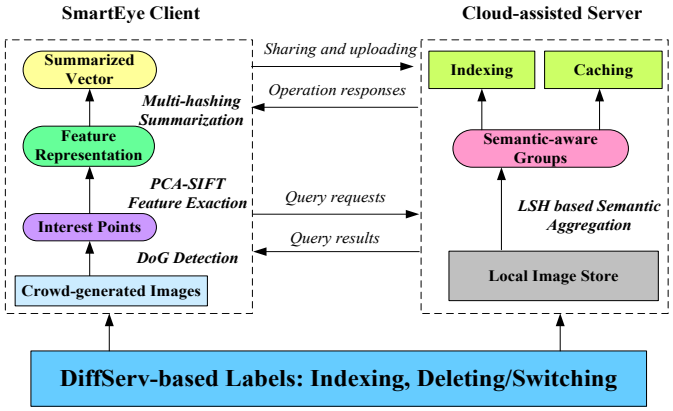


Fig. 2. The network communication between clients and servers.

run on the DiffServ labeling schemes via the mapping between similarity and labels.

SmartEye significantly cuts down the cost of sharing and uploading the crowd-generated images due to the salient features. Specifically, SmartEye only transmits a tiny fraction of all images, namely, the representative and unique images, in a near real-time manner. For the valuable images selected for transmission, SmartEye actually transmits unique ones, not existing in the cloud-assisted system, by means of the application-level similarity detection techniques. Moreover, for non-duplicate images, the interest-points based identification technique is used to support correlation-aware grouping and queries via LSH in the cloud-assisted system. Hence, similar images are found within one or a limited number of semantic-aware groups. SmartEye uses a top-k-based-feature set technique, to identify more similar images in the image data transmission.

C. The Clients in the Communications

The client consists of the modules of feature extraction and space-efficient summarization. The feature extraction module uses the DoG [5] and PCA-SIFT schemes [8] to respectively detect and represent interest points of an image. An interest point refers to the point that is stable under local and global perturbations in the image domain. Typical perturbations include deformations (e.g., affine transformations, scale changes, rotations and translations) illumination, and brightness variations. By capturing their interest points, SmartEye can identify and extract the features of similar images.

The functional modules allow SmartEye to significantly reduce the bandwidth consumption and decrease the computation complexity. SmartEye can use the feature extraction module to represent and detect the interest points in the similar images with the aid of the DoG scheme. The detected interest points are represented by the PCA-SIFT scheme in a compact way, thus obtaining substantial space savings. In order to obtain more space savings and efficiently support semantic grouping, SmartEye implements the summarization module in the context of DiffServ domain, in which we hash the features per image into a space-efficient Bloom-filter based indexing structure. Since similar images generally contain some identical features, these features can be projected to the same bits in the Bloom filters to represent the similarity of images.

D. The Cloud-assisted Servers

The main functions of cloud-assisted servers are to fast identify similar images and give operation responses to the clients, as well as system optimization, such as indexing, caching and prefetching. The servers leverage the semantic grouping module, in which SmartEye employs locality sensitive hashing (LSH) to efficiently capture correlated features that can accurately identify similar images. LSH's effectiveness and efficiency have been demonstrated in similarity search [16] and approximate queries [11], [17]. SmartEye uses the Bloom filters as the inputs to LSH in the semantic grouping module to aggregate correlated images together and support the mapping between the similarity identification in the servers and label management in the intermediate nodes.

The server end system maintains the image datasets from the clients over a bandwidth-constrained network. To avoid transmitting the entire crowd generated images, which result in an unacceptably long latency and high energy consumption in smartphones, the images from the representative image list identified by SmartEye are considered for online transmission.

To reduce the overhead of network bandwidth and reduce the energy consumption, we propose a feature-based top-k similarity scheme. This scheme is designed from an application's point of view, which allows SmartEye to identify more similar images to significantly reduce the number of images transmitted. Note that since the interest-point based local features of each image have been obtained when locally selecting the representative images, SmartEye can directly use these extracted features without recomputation.

E. The Workflow between Clients and Servers

The workflow of network transmission consists of procedures in both the client and the server as shown in Figure 3. For the source, ideally it needs to send the images as fast and as small bandwidth and energy as possible. For the cloud-assisted servers, they need to divide semantically correlated images into groups based on image metadata and content analysis to query the membership of similar images. The membership query of similar images helps determine if an image in the client needs to be transmitted.

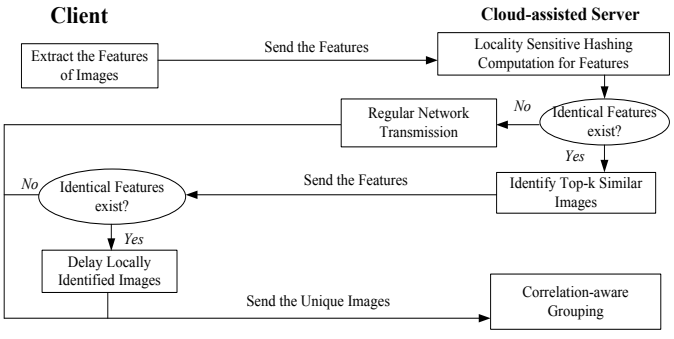


Fig. 3. The workflow of network transmission between clients and servers.

When the client needs to transmit an image, it first sends the interest-point based features to the cloud-assisted servers.

The cloud-assisted servers use LSH to hash similar images into the correlated groups. Within the narrowed search scope, SmartEye can easily and quickly determine the approximate membership of the image to be transmitted. A hit means that the cloud-assisted servers already contain images that are highly similar to the image targeted for transfer. SmartEye further checks if the existing image in the cloud-assisted servers contains richer content/metadata information (i.e., more representative) than the one to be transmitted. If so, the latter image does not need to be transmitted. Otherwise, this image, having richer information and being more representative (valuable), is transmitted to guarantee the analytics quality. Note that in the context of SmartEye, the images to be processed are allowed to be publicly accessed. Moreover, the similarity search in the correlation-aware groups will yield the top-k images with the most matching interest points to the queried interest points.

Near-duplicate images generally contain identical features. In the client where the image has not been deduplicated for near-duplicates, a feature corresponds to several images. The features of the identified top-k most similar images are sent to the source system to identify more similar images, further exploiting the content locality in the cloud storage. For example, besides the mentioned geometric transformation, there generally exist a large fraction of images that are semantically similar in content, e.g., taking pictures of the same scene from slightly different angles. It is very difficult to accurately identify such semantically similar images using the computer identification tools alone, due to the lack of accurate semantics identification techniques. Nevertheless, we believe intuitively that these pictures may share similar or identical features. The top-k similarity set has a large probability of covering these semantically similar images in contents.

IV. Implementation Details in the Cloud

In this section, we present the implementation details in both clients and cloud-assisted servers, including image feature extraction, semantic-aware aggregation and QoS-aware DiffServ design.

A. Image Features for Bandwidth-efficient Communications

SmartEye extracts the features of images as lightweight representation to support bandwidth-efficient communications. The features of an image have the advantages of being invariant to the scale and rotation of the image. The features can hence provide robust matching across a substantial range of changes. These changes include the affine distortion, the changes in various viewpoints, the additions of noise, and the changes in illumination. Moreover, we use interest points as an effective local descriptor, which have been widely employed and well-recognized in real-world applications. The typical applications include the object recognition and the image retrieval due to the benefits of being robust to photometric changes and geometric variation.

1) The Detection of Interest Points: In order to accurately extract and detect the interest points, we need to localize them in position and scale. In general, interest points exist at local peaks in a scale-space search. We can filter them to preserve those that possibly remain stable over transformations. Moreover, we select scale-space peaks and find potential interest points by scanning the image over location and scale. SmartEye executes these operations via building a Gaussian pyramid and searching for local peaks in a series of difference-of-Gaussian (DoG) images.

In order to efficiently identify and localize interest points, the potential points at each candidate location are localized to sub-pixel accuracy and eliminated. This occurs if exhibiting un-stability based on measures. Moreover, in order to examine the stability, we carry out the operation of orientation assignment. Based on local image gradient directions, we allocate one or more orientations to each interest point location. We transform the image via being relative to the assigned orientation, scale, and location for each feature.

2) The Representation of Interest Points: In order to efficiently represent the interest points, we construct the description of interest points to support fast retrieval performance. The description is distinctive, concise and invariant over transformations. The transformations are often caused by the changes in camera pose and lighting. In order to generate compact feature vectors, we need to compute the local gradient image of the patch, normalize it, and project it onto a pre-computed Eigen space. This Eigen space comes from a large number of interest points that are extracted from images of natural scenes. In practice, the number of descriptors depends on the cardinality of the set of detected interest points from an image.

B. Semantic-aware Aggregation

In order to identify similar images for the aggregated flows, we extract the features of images and use LSH to map them into the same or close-by hash buckets with a high probability [11]. In the meantime, Bloom-filter based vectors [12] serve as the input of LSH to obtain the space savings and decrease the complexity. Two Bloom filters representing two similar images contain a large fraction of the same bits.

A Bloom filter consists of a bit array of m bits for a dataset $S = \{a_1, a_2, \ldots, a_n\}$ of n items. All bits are initially set to 0. The Bloom filter contains k independent hash functions $\{f_1, \ldots, f_k\}$ to map data items to a bit vector $[1, \ldots, m]$. The hash function f_i is able to map an item a to one of the m-array bit positions in a uniformly-random manner. For a membership query, an item a is considered as the member of set S if all $f_i(a)$s are 1. Otherwise, it is not a member. A false positive may occur when an item a is mistakenly considered as a member although it in fact is not. The false positive is approximately $(1-e^{-\frac{kn}{m}})^k$ when the Bloom filter has m bits and k hash functions for a dataset with n items. The false-positive probability has a bound of $(1/2)^k$ or $(0.6185)^{m/n}$ when $k = (m/n)\ln 2$ [18].

LSH function families exhibit the locality-aware property, which support similarity aggregation. The similar images with identical features can be hashed into the same buckets with a higher probability. S is the domain and $||*||$ is the distance metric.

Definition 1: LSH function family, i.e., $\mathbb{H} = \{h : S \to U\}$, is called (R, cR, P_1, P_2)-sensitive for distance function $||*||$ if for any $p, q \in S$

- If $||p,q|| \leq R$ then $Pr_{\mathbb{H}}[h(p) = h(q)] \geq P_1$,
- If $||p,q|| > cR$ then $Pr_{\mathbb{H}}[h(p) = h(q)] \leq P_2$.

In the LSH, $c > 1$ and $P_1 > P_2$. In order to increase the gap between P_1 and P_2, SmartEye uses multiple hash functions. Distance functions $||*||$ are correlated with different LSH families of l_s norms based on an s-stable distribution. We hence obtain hash function $h_{a,b} : R^d \rightarrow Z$, which can map a d-dimensional vector v onto a set of integers. We further define hash function in $\mathbb{H}$ as $h_{a,b}(v) = \lfloor \frac{a \cdot v + b}{\omega} \rfloor$, in which a is a d-dimensional random vector. The vector can be chosen to the entries that follow an s-stable distribution. Moreover, b is a real number chosen uniformly from the range $[0, \omega)$.

C. QoS-aware DiffServ for In-network Deduplication

DiffServ can efficiently support in-network deduplication scheme in the context of SDN infrastructure. SmartEye executes complex operations, such as packet classification and similarity identification, at the edge of the network. The edge node labels the packet to obtain a particular type of per-hop behavior. Unlike the edge nodes, the core nodes only need to carry out simple operations, such as matching, forwarding or dropping, based on the semantics of PHBs. In the DiffServ domain, data are classified into different classes via different parameters, such as source address, destination address or the data similarity. SmartEye allocates DiffServ labels in the packets or chooses to overlook them. In the DiffServ, it is not necessary to setup the path and reserve end-to-end resources.

When clients need to upload images to the cloud, SmartEye extracts their features via SIFT tool. These features can represent the contents of images. SmartEye then transforms these features into the vectors of Bloom filters. The vectors are the input of LSH hash functions. SmartEye identifies and aggregates similar images into the semantic-aware groups via the LSH. We further allocate the same labels to the images from the same group. The aggregated flows with the same label will be routed into the same end-to-end path that consists of a series of SDN-enabled switches. Due to the similarity property in each aggregated flow, the images from one flow can be deduplicated with a high probability. The in-network deduplication can be completed in the switch that contains the frequently accessed data. If duplicate data are found, the corresponding flows will be removed.

V. Performance Evaluation

In this section, we present the experimental results of a SmartEye prototype implementation. We collect a large image set to comprehensively examine the performance of SmartEye with existing state-of-the-art solutions in several metrics.

A. Experiment Setup

The experiment implementation consists of two parts, i.e., the Android based client and the cloud-assisted servers. Specifically, in SmartEye's Android-based clients, we designed a friendly and easy-to-use interface for users to upload images and submit queries. To support local image processing, we ported an open-source implementation of PCA-SIFT feature extraction algorithm to Android. We also implemented the LSH algorithm, via summarized vectors, to convert from PCA-SIFT features to vectors. For clients, users need to download and install SmartEye's application software in their smartphones to execute the image identification and efficient network transmission in the context of SDN.

For the cloud-assisted servers, we implemented SmartEye on a 128-node cluster. Each node has a quad-core CPU running at 2.7GHz, with a 32GB RAM, a 500GB 7200RPM hard disk and a Gigabit network interface card. SmartEye leverages correlation-aware hashing techniques for fast and efficient image indexing, which contains real-world and large-scale image datasets.

1) Evaluation Workload: Real-world Image Sets: The used image sets come from the events of Typhoon Haiyan (2013) and Hurricane Sandy (2012). We collect real and openly assessable images from the popular image sharing websites and social networks, as well as the search engines for these two events, i.e., Typhoon Haiyan (10 million images) and Hurricane Sandy (12 million images).

SmartEye considers the spatial constraint with the zones affected by disasters. The unique and popular landmarks and sceneries are helpful for accurate and meaningful evaluation. We select 28 landmarks in the set of Typhoon Haiyan and 21 ones in the set of Hurricane Sandy. Hence, we only collect images that contain the corresponding representative landmarks. The collected image dataset ultimately contains 22 million images that require more than 60TB in the storage capacity. The main characteristics are shown in Table I.

TABLE I. The Properties of Collected Image Sets.

Dataset Name	No. Images	Total Size	File Type	Landmark
Haiyan	10 million	32.8 TB	bmp(6%), jpeg(85%), gif(9%)	28
Sandy	12 million	30.2 TB	bmp(5%), jpeg(80%), gif(15%)	21

2) Evaluation Baselines and Parameters: We compare SmartEye with the state-of-the-art schemes, SIFT [5], PCA-SIFT [8] and Multiple Feature Hashing (MFH) [19]. MFH maintains the local structure information of each individual feature. By using a group of hash functions, MFH builds the local structures for all the features. Hence, the video keyframes are mapped into the Hamming space to allow a series of binary codes to represent the image dataset. Moreover, in order to obtain appropriate R values in our experiments, we use the popular and well-recognized sampling method that was proposed in the original LSH [11] and has been used in real-world applications [17], [20]. By defining "*proximity measure* $\chi = ||p_1^\star - q||/||p_1 - q||$", we evaluate the top-1 query quality for queried point q. We respectively leverage $p_1^\star$ and p_1 to represent the actual and searched nearest neighbors of point q via distance computation. Finally, we obtain the suitable R values to be 750 and 950 respectively for the *Typhoon Haiyan* and *Hurricane Sandy* image datasets. In addition, we use $L = 8$, $\omega = 0.9$, $M = 12$ in the LSH-based computation and $k = 7$ for the hash functions in the Bloom filters.

In order to comprehensively evaluate the performance, we use multiple metrics, including query performance, space overhead, insertion latency and network transmission. For *Query Performance*, in each such query request, the expected results are images similar to the input portrait and the query latency is the time from when the request is submitted to when the results are returned. The query accuracy is affected by the potential false positives and false negatives. The false positives mean that an image dissimilar to the queried image is mistakenly

returned as a result. The false negatives mean that an image similar to the queried image fails to be identified. We randomly choose 5000 distinct images from the image datasets, which are considered concurrent query requests in the evaluation. In general, a very limited number of false positives generally have a subtle influence on the query results, which can be remedied easily by sparse checking manually in post-processing operations. As a result, we mainly report the query accuracy in terms of false negatives. For *Space Overhead*, the (near-) real-time processing scheme requires space-efficient storage structures. We examine the space overhead of all the evaluated index structures and check if the main memory with a limited size is able to contain the entire index structure to support (near-) real-time data analytics. For *Insertion Latency*, we investigate the average latency of inserting new images into an existing image dataset, respectively by the SmartEye, SIFT, PCA-SIFT and MFH schemes. The operation of inserting a batch of images entails first locating suitable grouping positions for the new images in the current dataset by considering their affinity and similarity to existing images and image groups there. This is then followed by the appropriate changes to the index structures so that the new images are fully indexed for future queries. For *Network Transmission*, we examine the bandwidth savings in clients, compared with full transmission for the images to be uploaded and shared.

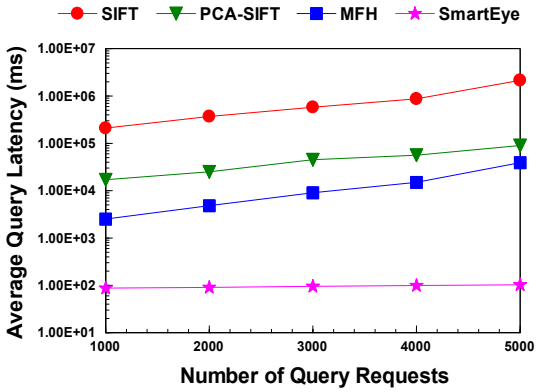
B. Results and Analysis

We present and analyze the evaluation results in terms of the performance metrics.

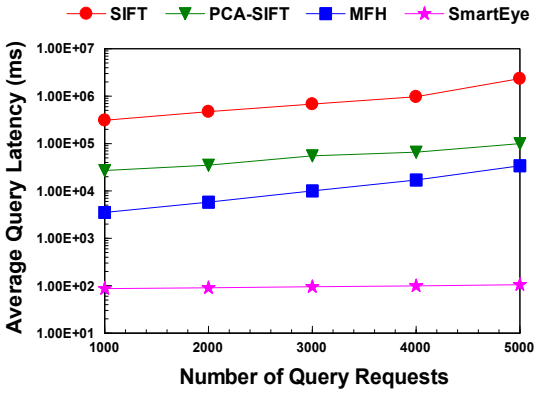
1) Query Latency: Figure 4 shows the average query latency. We examine the latency as a function of the number of query requests from 1000 to 5000 with the increase of 1000. The latency of PCA-SIFT, at 3.6min, is one order of magnitude better than SIFT's 52.5min, due to its PCA property. However, both SIFT and PCA-SIFT leverage linear-search matching to identify similar features. These features are generally stored in the SQL-based database. Due to the problem of space inefficiency, they have to frequently access the hard disks and thus incur more disk I/Os in the servers. Moreover, we observe that MFH performs better when the number of query requests is smaller (e.g., smaller than 1000). However, the performance decreases when the number of query requests increases. The main reason is that the MFH has to maintain the storage structure for each feature. The query latency of SmartEye is much shorter than the above schemes and obtains approximately at 127.9ms for all datasets. SmartEye hence obtains more than 3 orders of magnitude faster than PCA-SIFT and 2 orders of magnitude faster than MFH.

To further reduce the complexity, SmartEye uses the principal components analysis for dimensionality reduction to generate compact feature vectors, thus alleviating the space overhead. In the meantime, the semantic correlation exploits the advantages of low-complexity LSH using simple Bloom filter bit-array as the input. Due to the above design advantages, SmartEye obtains significant query performance improvements.

2) Query Accuracy: Table II shows the query accuracies of all evaluated schemes normalized to SIFT that is one of state-of-the-art exact-matching approaches and thus serves as the baseline, i.e., 100% accuracy in this metric.



(a) Typhoon Haiyan Dataset.



(b) Hurricane Sandy Dataset.

Fig. 4. The average query latency.

Since MFH leverages a series of binary codes to identify similar images, it causes the lowest accuracy, i.e., 92.98% average accuracy in the Typhoon Haiyan image dataset. Moreover, PCA-SIFT uses compact feature vectors via executing dimensionality reduction. Due to the decrease of the number of dimensions, PCA-SIFT obtains an accuracy of 99.9963%. The accuracy of SmartEye is 99.991%, which is slightly lower than that of PCA-SIFT. We conjecture that hash collisions in Bloom filters and LSH are error-prone in the DiffServ classification. The accuracy of SmartEye is 0.0053% lower than PCA-SIFT in Typhoon Haiyan image dataset. Furthermore, since SmartEye can obtain significant search-latency performance (by up to 3 orders of magnitude), the slight accuracy loss is acceptable in the context of disaster relief environments.

TABLE II. THE QUERY ACCURACY NORMALIZED TO SIFT.

Image Sets	No. Queries	SIFT	PCA-SIFT	MFH	SmartEye
Haiyan	1000	100%	99.9988%	95.1%	99.997%
	2000	100%	99.9972%	94.3%	99.995%
	3000	100%	99.9968%	93.2%	99.991%
	4000	100%	99.9951%	91.7%	99.988%
	5000	100%	99.9939%	90.6%	99.982%
Sandy	1000	100%	99.9982%	94.4%	99.992%
	2000	100%	99.9965%	93.6%	99.986%
	3000	100%	99.9959%	92.9%	99.981%
	4000	100%	99.9945%	91.7%	99.974%
	5000	100%	99.9927%	90.1%	99.968%

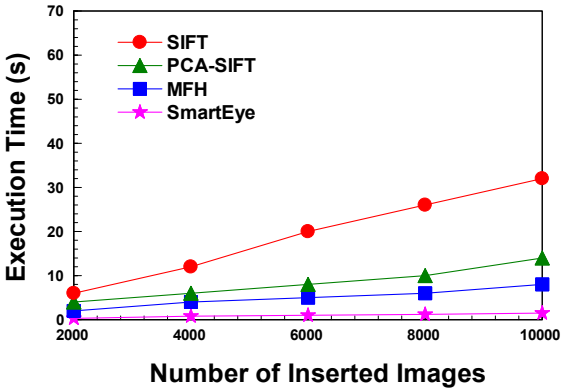
3) Space Overhead: Table III shows the space overheads of SIFT, PCA-SIFT, MFH and SmartEye, normalized to that of SIFT. Due to the decrease of the number of dimensions, PCA-

SIFT can obtain space savings of about 20% over SIFT. Since MFH leverages a series of binary codes in the Hamming space, it requires around half of storage space of SIFT. Moreover, SmartEye only keeps the summary vectors of the features, it can save space overhead by around 10% of the space overhead required by SIFT. Therefore, SmartEye can further improve the query performance by placing more index information into the main memory.

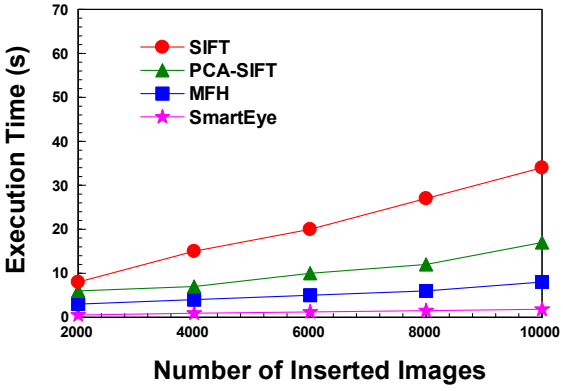
TABLE III. SPACE OVERHEAD NORMALIZED TO SIFT.

Image Datasets	SIFT	PCA-SIFT	MFH	SmartEye
Typhoon Haiyan	1	0.87	0.62	0.15
Hurricane Sandy	1	0.79	0.53	0.12

4) The Latency of Inserting Data: Figure 5 shows the latency of inserting images by using the SmartEye scheme. For the Typhoon Haiyan image dataset, inserting 10,000 new images requires 32.2s in SIFT, 15.9s in PCA-SIFT, 4.3s in MFH and 1.8s in SmartEye. The advantage of SmartEye comes from the accurate location of new images via feature summary vectors and semantic aggregation. Moreover, with the increase of the number of inserted images, the latencies of SIFT and PCA-SIFT show the growth in a linear manner. Unlike them, SmartEye's insertion latency is still flat due to the use of $O(1)$ complexity LSH. For each image, MFH needs to map the video keyframes into the codes in the Hamming space. For the Hurricane Sandy image dataset, we observe the similar results.



(a) Typhoon Haiyan Dataset.



(b) Hurricane Sandy Dataset.

Fig. 5. The latency of inserting images.

5) Network Transmission: A smartphone uploads all images to be transmitted to the destination server and requires continuous bandwidth guarantee, which is difficult to offer in a crowdsourcing environment. SmartEye leverages near-duplicate identification technique to significantly reduce the amounts of images to be transmitted. We compare SmartEye with the Chunk-based scheme due to its energy efficiency, which has been examined and recommended by the evaluation of battery power consumption with 11 Internet applications [21].

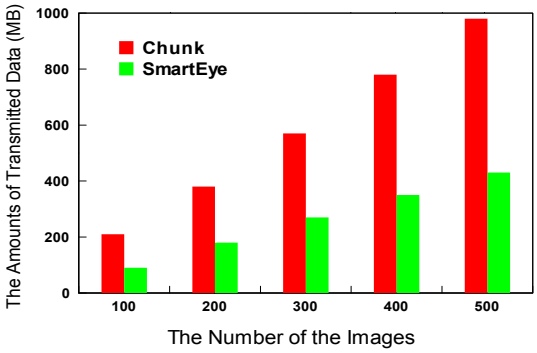


Fig. 6. Network transmission overhead.

Figure 6 shows transmission overhead. We evaluate the bandwidth performance of transmitting a batch of images. Unlike chunk-based transmission scheme, SmartEye can obtain more than 52.7% bandwidth savings, since it reduces the number images to be transmitted. Moreover, the percentage of bandwidth savings increases with the growth of the number of images, since more images contain similar ones with a higher probability. SmartEye hence shows the potential benefits in terms of network communication due to the suitable use of QoS-aware DiffServ scheme to efficiently support the in-network deduplication.

VI. RELATED WORK

Community sense and response systems from Caltech [22] gather and share human-generated contents from users' Internet-enabled devices for real-time awareness of dangerous earthquakes. CrowdSearch [23] combines automated image search with real-time human validation of search results. Automated image search is performed using a combination of local processing on mobile phones and backend processing on remote servers. Due to the needs of massive human responses, the searching incurs long latency and become error-prone. To alleviate energy waste in wireless video streaming, a download scheduling algorithm, called eSchedule [24], is proposed to predict user behavior by leveraging crowd-sourced video viewing statistics. 11 Internet streaming applications are examined in [21] using different streaming protocols to empirically investigate the battery power consumption on the wireless network interface. Carroll et al. [25] present a detailed power consumption analysis of different smartphone subsystems based on measurements of a physical device. TailEnder [26] is a protocol that reduces energy consumption of common mobile applications. Cells [27] is a virtualization architecture for enabling multiple virtual smartphones to run simultaneously on the same physical cellphone in an isolated, secure manner. SmartEye leverages near-duplicate image detection to improve bandwidth efficiency in network communications and obtain energy savings in smartphones.

DDFS [28] presents the scheme of exploiting the backup-stream locality to reduce network bandwidth and alleviate the frequent accesses to on-disk index. Moreover, LBFS [29] identifies the similarities between files or versions of the same file to deliver high performance in a low-bandwidth network file system. EndRE [30] leverages a fingerprinting index scheme to opportunistically use system resources on end hosts and obtain similarity compression gains. In contrast to these existing system-level approaches, SmartEye provides both application-level and system-level detection for both identical and near duplicate data.

VII. Conclusion

In order to improve the performance of network transmission, we propose the in-network deduplication scheme, in which the deduplication can be completed in the intermediate nodes, rather than the conventional source or destination end systems. Due to the fast operation response and significant bandwidth savings, the proposed SmartEye can efficiently support the image retrieval in the context of disaster relief. To implement the SmartEye, we build the in-network deduplication scheme over the QoS-aware DiffServ architecture. We hence carry out the differentiated deduplication services based on the similarity of images, which are accurately and efficiently identified by semantic hashing. The similarity can be mapped into the labels in the DiffServ, which presents the different PHBs. SmartEye can identify the unique and important images in an efficient manner. The remaining images will be temporarily stored in local smartphones, thus significantly reducing the bandwidth and energy consumption for image sharing.

In essence, SmartEye bridges the gap between data communications in the network and deduplication systems in the storage, which are often studied separately. To the best of our knowledge, SmartEye is the first work of implementing the in-network deduplication scheme over the DiffServ architecture. SmartEye is demonstrated to be a useful tool in supporting near real-time processing of real-world cloud-assisted disaster environments.

Acknowledgements

This work was supported in part by National Natural Science Foundation of China (NSFC) under Grant 61173043, National Basic Research 973 Program of China under Grant 2011CB302301, NSERC Discovery Grant 341823 and US National Science Foundation Award 1116606.

References

[1] [New Scientist] Social media helps aid efforts after typhoon Haiyan *http://www.newscientist.com/article/dn24565-social-media-helps-aid-efforts-after-typhoon-haiyan.html#.UoQ2FtKOS8c*.

[2] J. Gantz and D. Reinsel, "The Digital Universe in 2020: Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East," *International Data Corporation (IDC) iView*, December 2012.

[3] Y. Hua, B. Xiao, and X. Liu, "NEST: Locality-aware Approximate Query Service for Cloud Computing," *Proc. IEEE International Conference on Computer Communications (INFOCOM)*, 2013.

[4] Y. Ke, R. Sukthankar, and L. Huston, "Efficient near-duplicate detection and sub-image retrieval," *Proc. ACM Multimedia*, 2004.

[5] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[6] H. Kim, N. Agrawal, and C. Ungureanu, "Revisiting storage for smartphones," *ACM Transactions on Storage*, vol. 8, no. 4, 2012.

[7] Y. Hua and X. Liu, "Scheduling heterogeneous flows with delay-aware deduplication for avionics applications," *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, vol. 23, no. 9, pp. 1790–1802, September 2012.

[8] Y. Ke and R. Sukthankar, "PCA-SIFT: A more distinctive representation for local image descriptors," *Proc. CVPR*, 2004.

[9] J. Liu, Z. Huang, H. T. Shen, H. Cheng, and Y. Chen, "Presenting Diverse Location Views with Real-time Near-duplicate Photo Elimination," *Proc. ICDE*, 2013.

[10] Changewave research *http://www.changewaveresearch.com, 2011*.

[11] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," *Proc. STOC*, 1998.

[12] B. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[13] Y. Hua, B. Xiao, D. Feng, and B. Yu, "Bounded LSH for Similarity Search in Peer-to-Peer File Systems," *Proc. ICPP*, pp. 644–651, 2008.

[14] F. Ribeiro, D. Florencio, and V. Nascimento, "Crowdsourcing subjective image quality evaluation," in *Proc. IEEE ICIP*, 2011.

[15] Y. Hua, X. Liu, and D. Feng, "Smart In-Network Deduplication for Storage-aware SDN," *Proceedings of ACM SIGCOMM*, pp. 509–510, 2013.

[16] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li, "Multi-probe LSH: Efficient indexing for high-dimensional similarity search," *Proc. VLDB*, pp. 950–961, 2007.

[17] Y. Hua, B. Xiao, B. Veeravalli, and D. Feng, "Locality-Sensitive Bloom Filter for Approximate Membership Query," *IEEE Transactions on Computers*, vol. 61, no. 6, pp. 817–830, 2012.

[18] A. Broder and M. Mitzenmacher, "Network applications of Bloom filters: a survey," *Internet Mathematics*, vol. 1, pp. 485–509, 2005.

[19] J. Song, Y. Yang, Z. Huang, H. T. Shen, and R. Hong, "Multiple feature hashing for real-time large scale near-duplicate video retrieval," *Proc. MM*, 2011.

[20] Y. Tao, K. Yi, C. Sheng, and P. Kalnis, "Quality and Efficiency in High-dimensional Nearest Neighbor Search," *Proc. SIGMOD*, 2009.

[21] Y. Liu, L. Guo, F. Li, and S. Chen, "An empirical evaluation of battery power consumption for streaming data transmission to mobile devices," in *Proc. Multimedia*, pp. 473–482, 2011.

[22] M. Faulkner, R. Clayton, T. Heaton, K. M. Chandy, M. Kohler, J. Bunn, R. Guy, A. Liu, M. Olson, M. Cheng, *et al.*, "Community sense and response systems: Your phone as quake detector," *Communications of the ACM*, vol. 57, no. 7, pp. 66–75, 2014.

[23] T. Yan, V. Kumar, and D. Ganesan, "Crowdsearch: exploiting crowds for accurate real-time image search on mobile phones," *Proc. MobiSys*, 2010.

[24] M. A. Hoque, M. Siekkinen, and J. K. Nurminen, "Using crowd-sourced viewing statistics to save energy in wireless video streaming," *Proc. MobiCom*, pp. 377–388, 2013.

[25] A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone," in *Proc. USENIX ATC*, 2010.

[26] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: a measurement study and implications for network applications," *Proc. ACM IMC*, 2009.

[27] J. Andrus, C. Dall, A. V. Hof, O. Laadan, and J. Nieh, "Cells: a virtual mobile smartphone architecture," *Proc. ACM SOSP*, 2011.

[28] B. Zhu, K. Li, and H. Patterson, "Avoiding the disk bottleneck in the data domain deduplication file system," *Proc. FAST*, 2008.

[29] A. Muthitacharoen, B. Chen, and D. Mazieres, "A low-bandwidth network file system," *Proc. SOSP*, 2001.

[30] B. Aggarwal, A. Akella, A. Anand, A. Balachandran, P. Chitnis, C. Muthukrishnan, R. Ramjee, and G. Varghese, "EndRE: an end-system redundancy elimination service for enterprises," *Proc. NSDI*, 2010.