Managing Redundant Content in Bandwidth Constrained Wireless Networks

Tuan Dao, Amit K. Roy-Chowdhury, Srikanth V. Krishnamurthy
U.C. Riverside

Harsha V. Madhyastha
University of Michigan

Tom La Porta
Penn State University

Premise

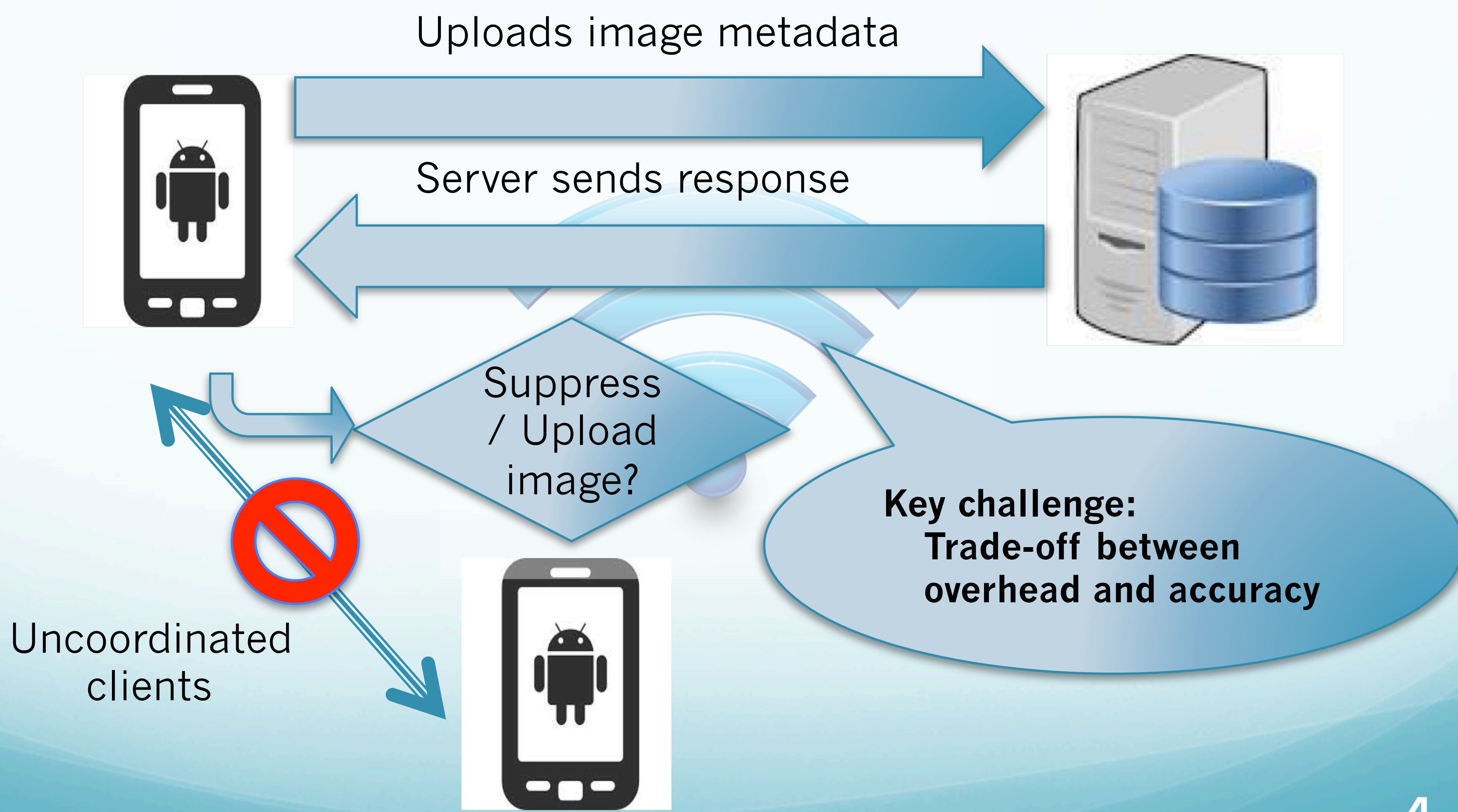
- During natural disasters, people tend to upload redundant images.
- Strained wireless network becomes congested
- *Suppressing* redundant content helps reduce latency in uploading *unique* and *critical* content.

Redundancies in other scenarios

- Similar images also uploaded in other scenarios
  - Examples: Concerts, sport events, etc.
- Redundant images can be lazily uploaded (when network is less congested)
  - Reduce network load and speed up transfers

How we manage redundant content

Goal: Accurately and efficiently determine if an image to be uploaded has similar version(s) on server



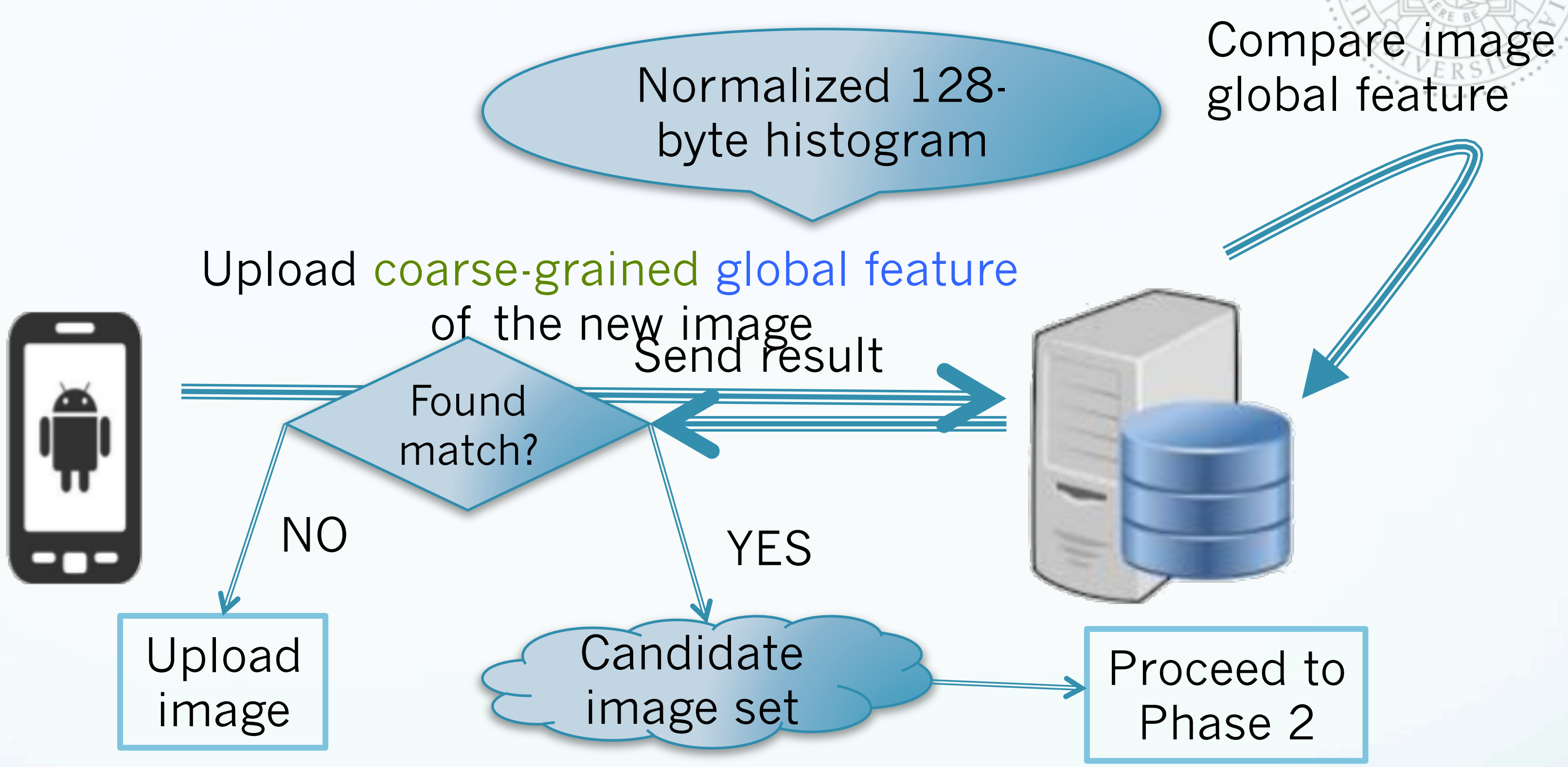
Roadmap

- Introduction
- **Detection of redundancies in images**
- Evaluation
- Conclusions

Overview of our approach

- 3-phase hierarchical approach:
  - Metadata overhead increases with every phase
  - Proceed to next phase only if classification decision cannot be made

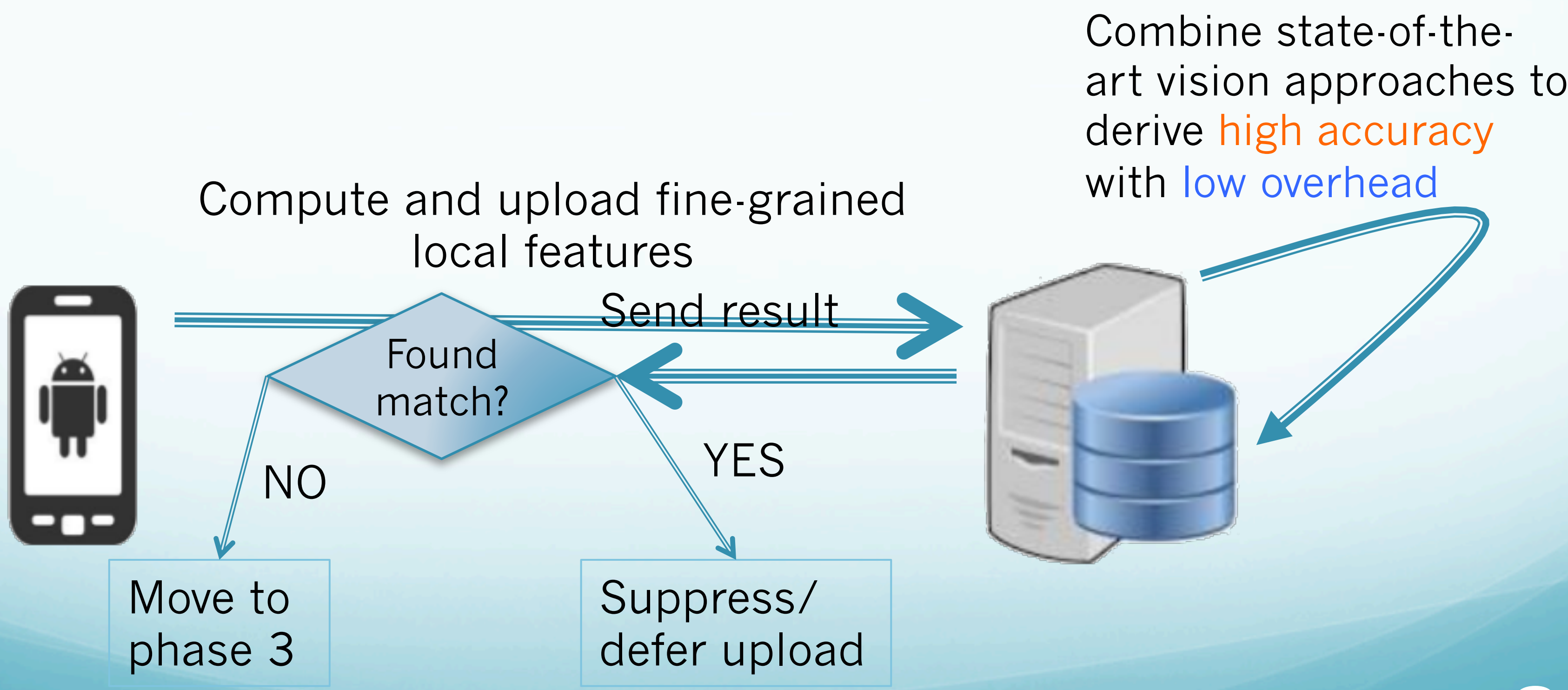
Phase 1: Use global color histogram



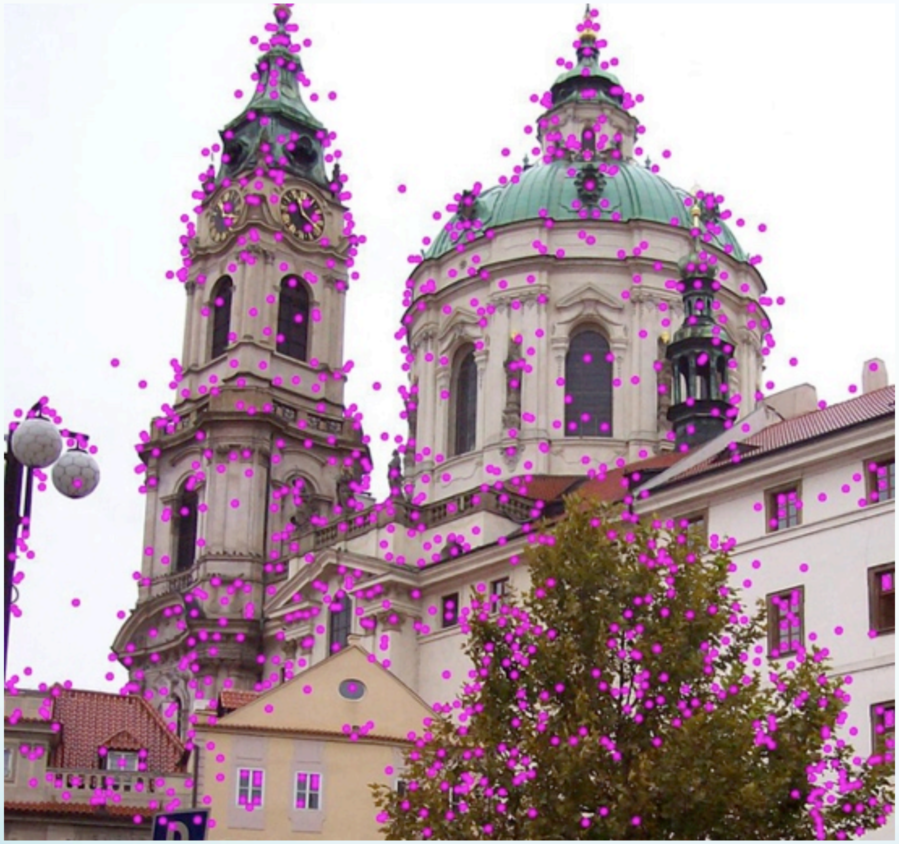
- Fast and easy to calculate (low overhead)
- Effective in determining dissimilar images, but high false positive rate

Phase 2: Use local features

- Use image local features to reduce false positives
- Choosing local features: Trade-off between detection accuracy and overhead



Using image key-points to detect similarity



- Key-points are distinctive patches (*local* features)
- Use ORB algorithm to extract image key-points
  - Two orders of magnitude faster than SIFT
  - Comparable results
- Data to store key-points larger than image content

  → Cannot compare image key-points directly

Image source: http://en.wikipedia.org/wiki/Scale-invariant_feature_transform

Compact representation of local features

- Bag-of-word (BoW) representation:
  - Represent each image as a histogram of visual words

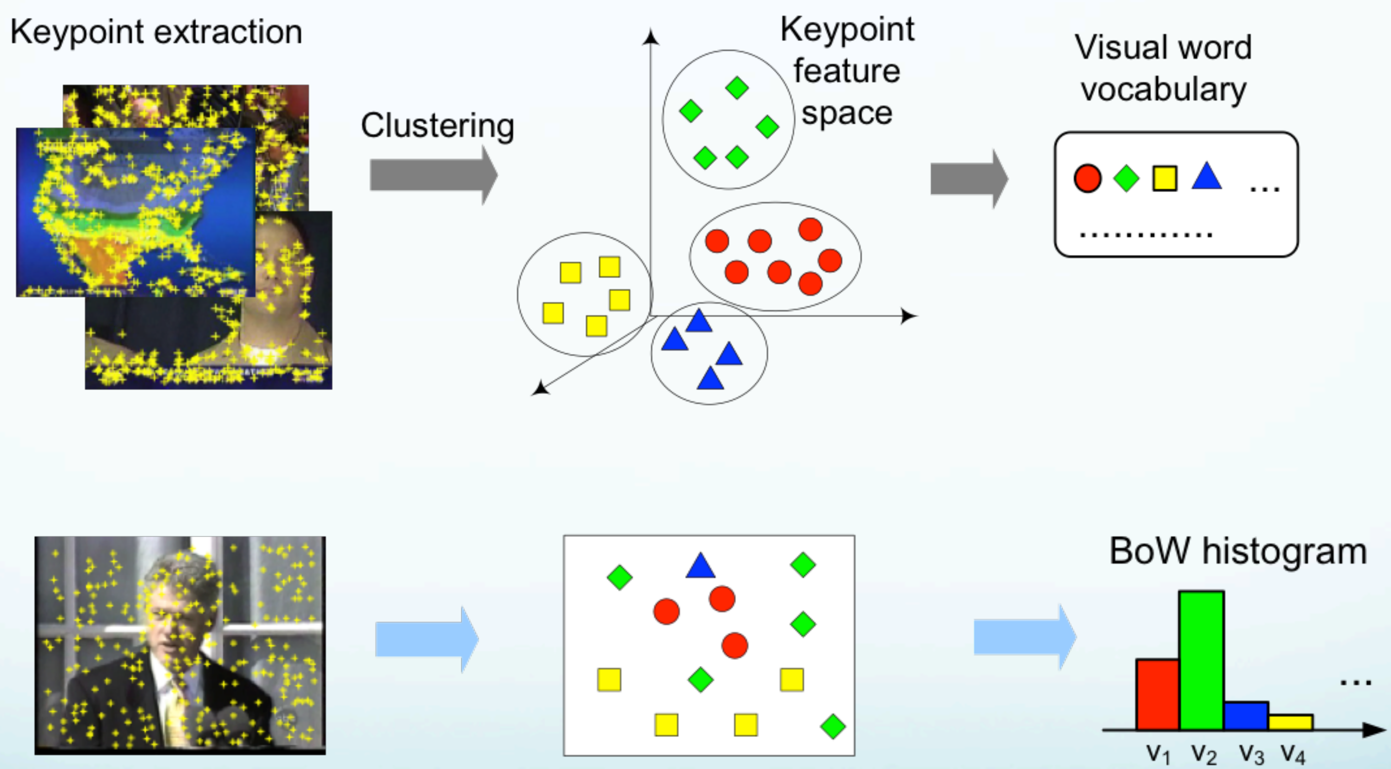


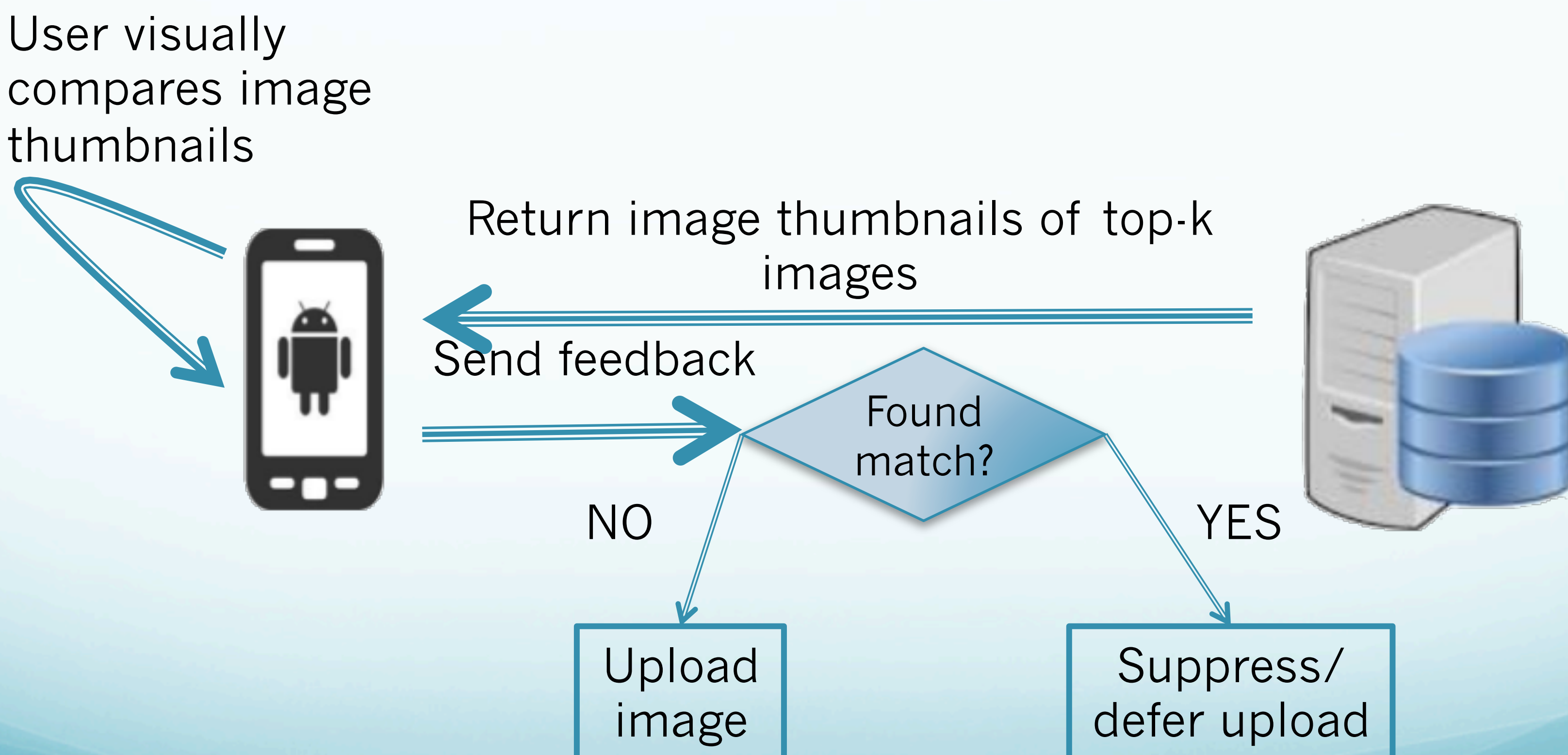
Image source: DVMM Lab, Columbia University

Using min-hash to approximate similarity

- Min-hash function: hashes a BoW representation into 1 number
  - *Assign* each visual word W a unique value h(W)
  - Min-hash of image $\mathbf{I}$: m($\mathbf{I}$) = min { h(W), W ε $\mathbf{I}$ }
  - If k equal min-hashes among n total generated min-hashes:
    similarity $(\boldsymbol{I}, \boldsymbol{I'}) \approx k/n$
- Embed geometric information (position in the image) of key-points to reduce *false positive* rate
- Fine-grained information sent to server in phase 2:
  - *k* min-hash values
  - geometric information of *k* *corresponding* key-points

Phase 3: Solicit user's feedback

- Phase 2 helps achieve very low false positive rate
- Phase 3, which solicits user feedback, boosts true positive rate



Summary of approach

① Upload coarse-grained global features for quick assessment

- 128-byte color histogram

② Upload fine-grained local features to reduce *false negatives*

- 1024 bytes for 512 min-hash values
- For each min-hash, 1 byte for geo-information of the corresponding key-point

③ Solicit user feedback to improve *true positives*

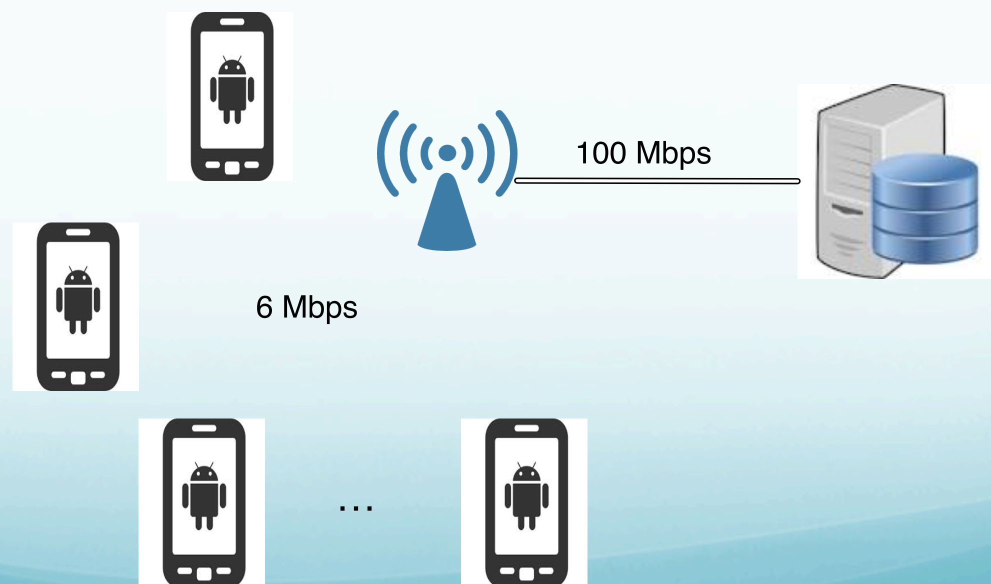
- 5 * 15 KB per thumbnail

Roadmap

- Introduction
- Detection of redundancies in images
- **Evaluation**
- Conclusions

Evaluation overview

- Conducted experiments to demonstrate
  - Detection accuracy: *True Positive* & *False Positive* rates
  - Impact on network performance: reduction in delays and network load
- System setup:
  - 20 Android phone testbed
  - Phones connect to central server through shared WiFi network



Testing and Training datasets

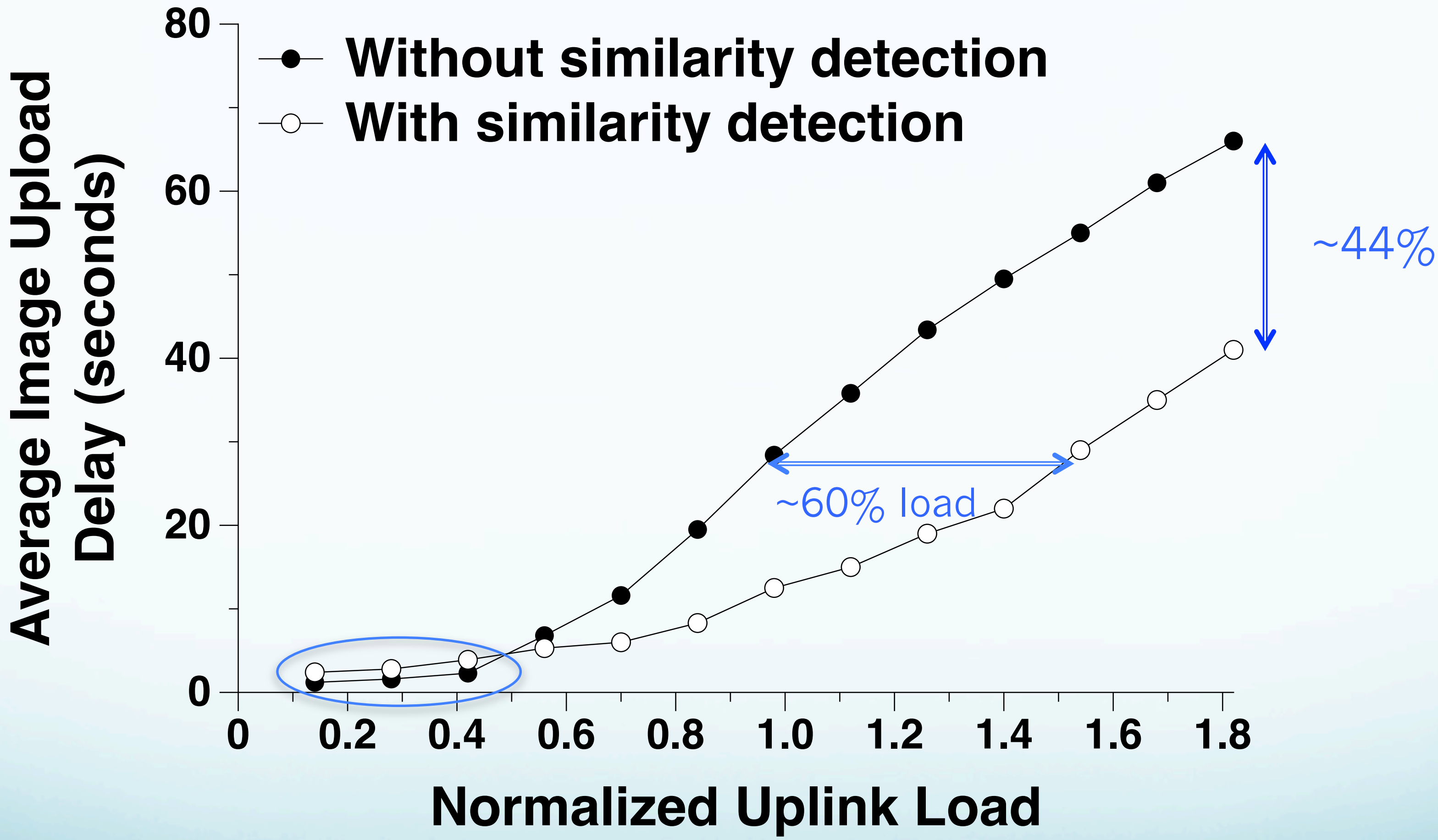
- Two image datasets with *ground-truth* information
  - University of Kentucky image set: 10,200 images, consists of 2,550 groups of similar images
  - "US cities" image set: 5,000 images, each from a different US city
- Evaluation setup:
  - Phones in testbed upload 1,000 images each from either dataset
  - Remaining images are pre-uploaded to server

One image group in the **Kentucky** dataset

Detection accuracy

Method	True Positive rate	False Positive rate
1 - Histogram matching	80%	65%
2 - Local features matching	50%	1%
3 – Feedback based on thumbnails		
• 1 thumbnail	59%	1%
• 3 thumbnails	64%	
• 5 thumbnails	69%	
• 10 thumbnails	71%	

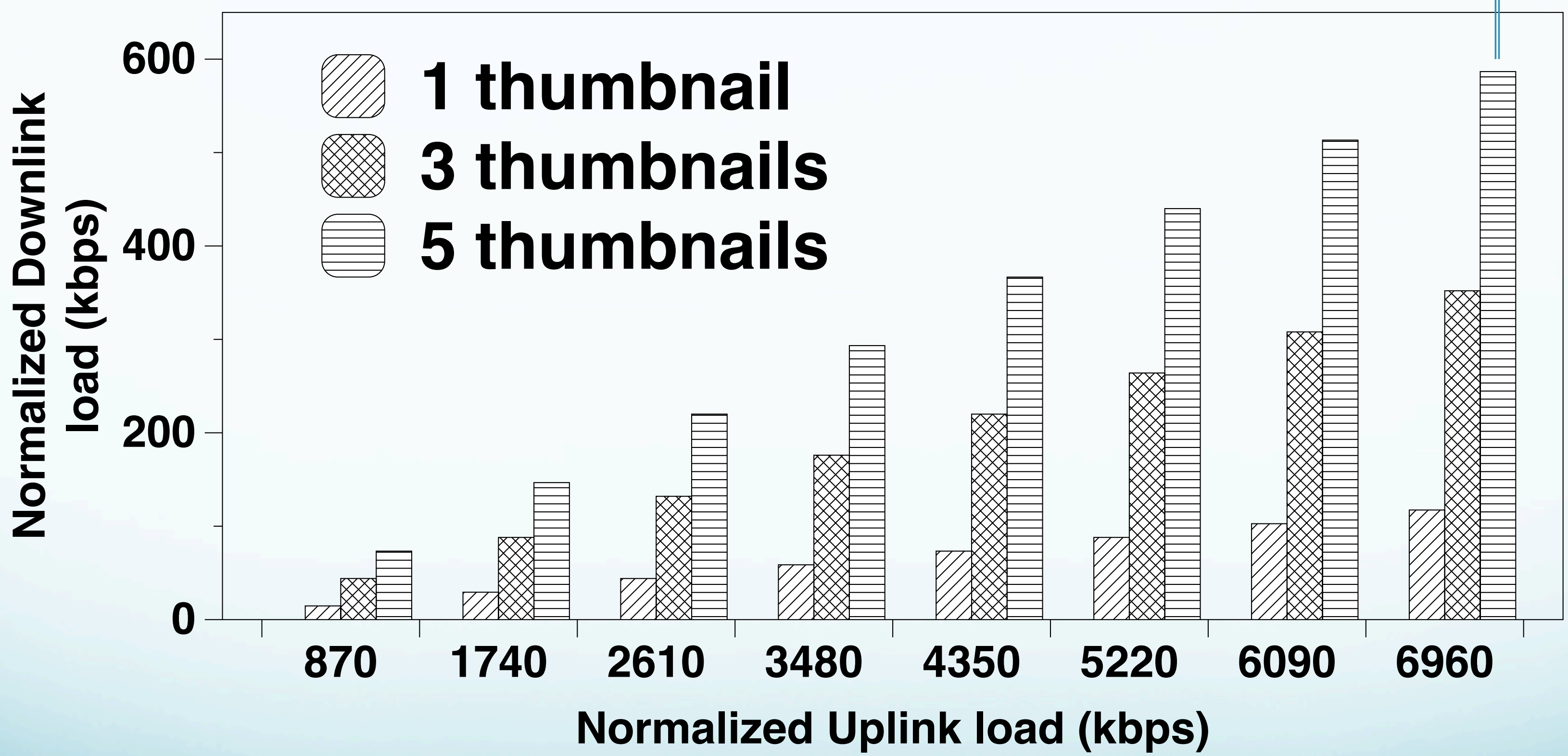
Impact on network performance



True Positive rate: ~70%; 50% similar images

Overhead of generating thumbnails

<10% of the link capacity



Link capacity: 6 Mbps

Conclusions

- Framework for detection of similar images when uncoordinated set of clients uploading to common server
- Leverage, but *intelligently* combine, many state-of-the-art vision algorithms to effectively detect similar images
- Experiments on phone testbed *(and using ns3 simulations)* to demonstrate impact on increasing network performance
- Future work:
  - Leverage device features (GPS location, camera orientation)
  - Take into account image priority, e.g. resolution, coverage, etc.

Thank you!

Tuan Dao - tdao006@cs.ucr.edu